ＡＤＳＰシリーズ

インストールガイド

ＭＡＴＬＡＢ対応 LIB

中部電機株式会社

目　　次

１．概要

本書は、ＭＡＴＬＡＢ対応ライブラリ（マルチ）「ＡＤＳＰ３２４－４１Ｄ／ＡＤＳＰ６７４－３４１Ｄ」及び、ＭＡＴＬＡＢ対応ＵＩＬ　「ＡＤＳＰ３２４－４９Ｄ／ＡＤＳＰ６７４－３４９Ｄ」及び、モデルコントロール　「ＡＤＳＰ３２４－５０／ＡＤＳＰ６７４－３５０」のインストール方法について説明しています。本書が説明している内容は、ＤＳＰのセットアップを含めた全体の手順概略と、本製品の具体的インストール手順です。ＤＳＰのセットアップ、コンパイラーのセットアップ等については、個々の製品の説明書が必要になりますので、本製品以外に以下の製品の説明書等をお手元に用意してください。

ＡＤＳＰ３２４－４１Ｄ／ＡＤＳＰ３２４－４９Ｄ／ＡＤＳＰ３２４－５０の場合

○ ＤＳＰボード「ＡＤＳＰ３２４－００Ａ スタートアップガイド」
○「ＡＤＳＰ３２４－４１Ｄ ライセンスシート」
○「ＡＤＳＰ３２４－４９Ｄ ライセンスシート」
○「ＡＤＳＰ３２４－５０ ライセンスシート」
○ ＭＡＴＬＡＢ対応ライブラリ・ユーザーズマニュアル（324-7x）ＰＤＦファイル
○ＤＳＰ用コンパイラ ＣＰＬ３２０ＦＬＴ－ＭＳ の説明書
○ ＭＡＴＬＡＢ／Ｓｉｍｕｌｉｎｋ／ＲＴＷの説明書

ＡＤＳＰ６７４－３４１Ｄ／ＡＤＳＰ６７４－３４９Ｄ／ＡＤＳＰ６７４－３５０の場合

○ ＤＳＰボード「ＡＤＳＰ６７４－００ スタートアップガイド」
○「ＡＤＳＰ６７４－３４１Ｄ ライセンスシート」
○「ＡＤＳＰ６７４－３４９Ｄ ライセンスシート」
○「ＡＤＳＰ６７４－３５０ ライセンスシート」
○ ＭＡＴＬＡＢ対応ライブラリ・ユーザーズマニュアル（674-7x）ＰＤＦファイル
○ＤＳＰ用コンパイラ ＣＰＬ３２０ＦＬＴ－ＭＳ の説明書
○ ＭＡＴＬＡＢ／Ｓｉｍｕｌｉｎｋ／ＲＴＷの説明書

注）ＰＤＦファイルをお読みいただくには[1]adobe Readerが必要となります。下記HPよりダウンロードしていただきご利用お願いいたします。

ダウンロードアドレス　http://www.adobe.co.jp/products/acrobat/readstep2.html

1) Adobe、Adobe ロゴ、Readerは、Adobe Systems Incorporated（アドビシステムズ社）の米国ならびに他の国における商標または登録商標です。

- MATLAB, SIMULINK, Real-Time Workshop, は米国 The Math Works Inc. の製品です。
- コンパイラ CCS は Texas Instruments 社の製品です。

２．全体の手順

本製品が使える状態までセットアップする手順（及び参照する説明書）は以下の通りです。

１、 DSP と DSP 用コンパイラセットアップ（スタートアップガイド）

２、 MATLAB のインストール（MATLAB の説明書、及び本書）

３、 旧版の MATLAB 対応ライブラリのアンインストール

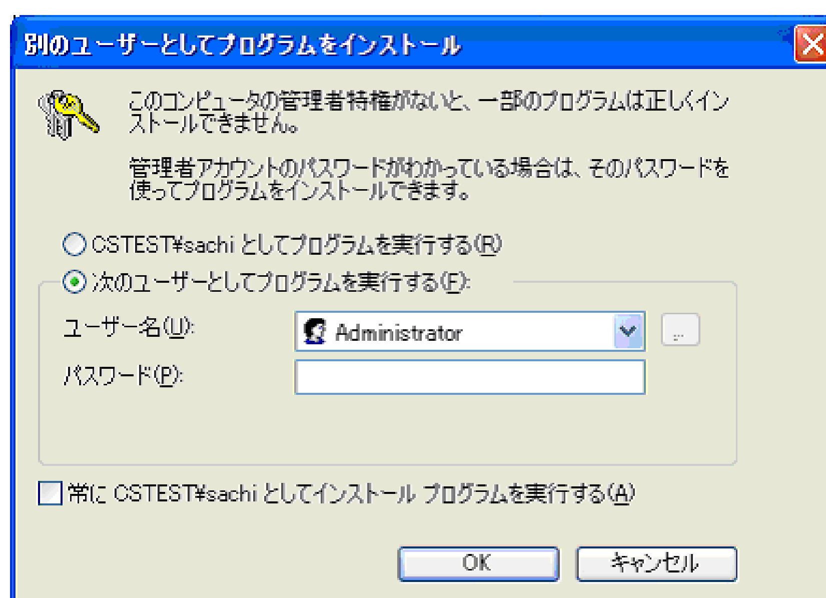
４、 ADSPXX4-x41D／x49D／x50 のインストール（本書）

１～２項については、各資料を参照のうえ、セットアップを行ってください。
３～４項については、次ページ以後の説明に従ってください。

“２、ＭＡＴＬＡＢインストール後”ＭＡＴＬＡＢを起動し以下の設定を確認して下さい。
メニューバーより→ファイル→設定→「一般」→“Toolbox ｷｬｯｼｭを行う”のﾁｪｯｸをはずす。
設定後一旦ＭＡＴＬＡＢを閉じてから“３、”以降の処理へ移ってください。

≪注意≫
インストールは、管理者権限を持つユーザにて実行してください。管理者権限のないユーザにて実行した場合は正常にインストールされませんので、一度アンインストールしてから再度インストールを実行してください。
管理者権限のないユーザにてインストールを開始すると以下のメッセージが表示される場合があります。ユーザの切り替えをして、再度インストールをしてください。

インストール先フォルダは、名称に日本語・空白を含まないフォルダへインストールしてください。例えば、”C:¥program files”の下へインストールするのは NG です。



３．インストール手順

３．１．　ＤＳＰとＤＳＰ用コンパイラセットアップ

ＤＳＰﾎﾞｰﾄﾞに添付している「スタートアップガイド」及び、ＤＳＰ用コンパイラ（ＣＰＬ３２０ＦＬＴ－ＭＳ／ＣＰＬ３２０６Ｘ－ＭＳ／ＨＬＬ３２０６Ｘ－ＭＳ）の説明書を参照し、ＤＳＰのセットアップとコンパイラーのインストールからＤＳＰボードの動作確認まで行ってください。

既にインストールしてある場合は、再インストールは不要です。

３．２．　MATLAB のインストール

使用出来る MATLAB は、7.1(R14 SP3)版です。既にインストールしてある場合は、再インストールは不要です。
それ以外の版をお使いの場合や、まだ MATLAB がインストールされていない場合は、MATLAB の取扱説明書を参照のうえ MATLAB のインストールを行ってください。

３．３．　旧版 MATLAB 対応ライブラリのアンインストール

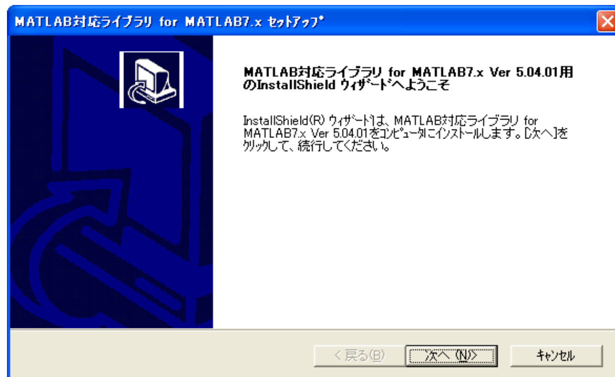
旧版の弊社 MATLAB 対応ライブラリ「ADSPxx4-x41D/x49D/x50」をご利用の場合は、それらの製品をアンインストールしてください。

アンインストールは、Windows のコントロールパネルにある、「プログラムの追加と削除」を起動し、「MATLAB 対応ライブラリ　for MATLAB7.x Ver x.xx.xx」の項目を選択し、「変更と削除」ボタンで削除します。
　補足：上記の x.xx.xx は MATLAB 対応ライブラリのバージョンを表します。

３．４．　MATLAB 対応ライブラリのインストール

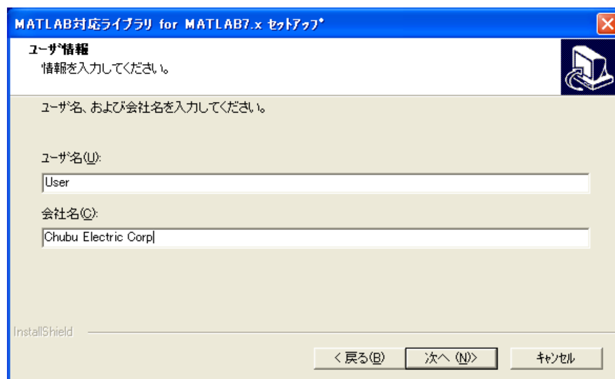
最後に本製品をインストールします。
CD-ROM を CD ドライブに挿入するとインストーラが自動的に起動します。起動しない場合は、エクスプローラ等で CD の¥Setup.exe を起動します。

【１】ようこそ



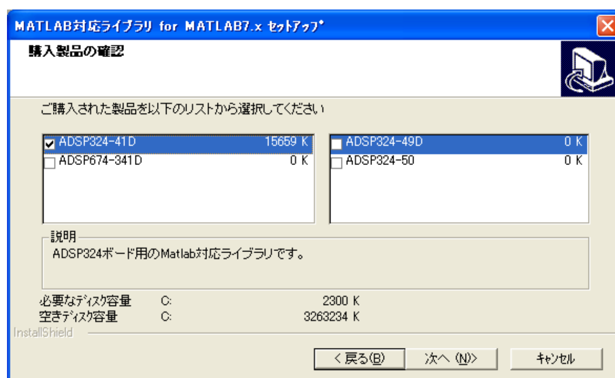
この画面が出たら、次へ(N) ボタンを押してください。

【２】ユーザ情報の入力



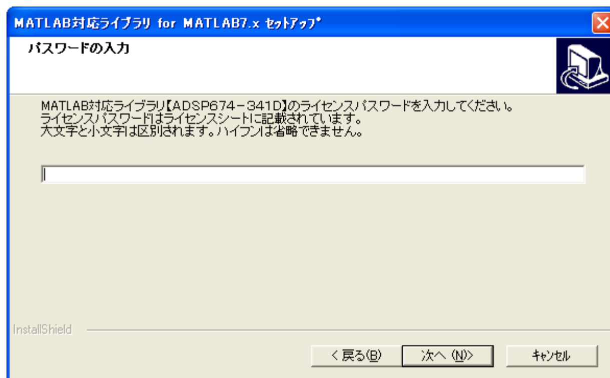
ユーザ情報を入力し次へ(N)ボタンを押してください。

【３】購入製品の確認



購入製品を確認し、チェックしてください。

【４】MLLIB のライセンスコードの入力



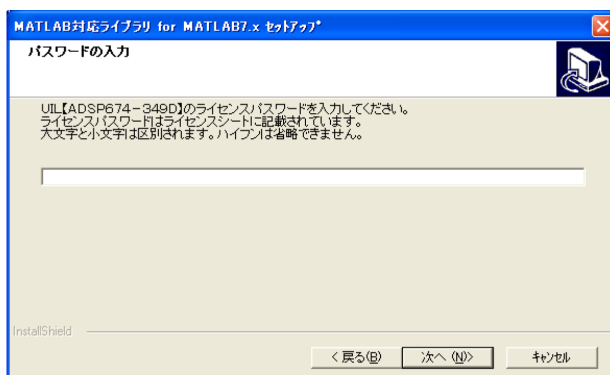
次に、“MLLIB のライセンスコードの入力”画面が出ます。

MATLAB 対応ライブラリ（ADSPxx4-x41D）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている － 記号も省略出来ません。

正しく入力したら、次へ(N) ボタンを押してください。

【５】UIL のパスワードの入力



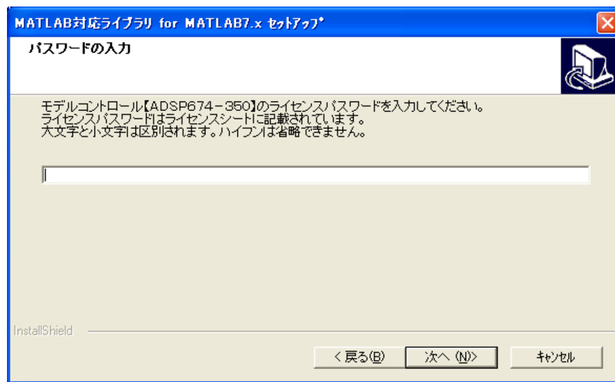
UIL を購入された場合のみ、“UIL のパスワードの入力”画面が出ます。

MATLAB 対応 UIL（ADSPxx4-x49D）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている － 記号も省略出来ません。

正しく入力したら、次へ(N) ボタンを押してください。

【６】モデルコントロールのパスワードの入力



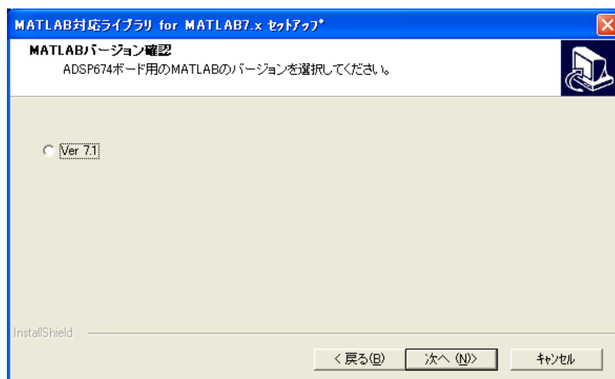
モデルコントロールを購入された場合のみ、“モデルコントロールのパスワードの入力”画面が出ます。

MATLAB 対応モデルコントロール（ADSPxx4-x50）のライセンスシートに記載されているパスワードを入力してください。

パスワードの大文字小文字は区別されます。正確に入力してください。また４桁毎に入っている－ 記号も省略出来ません。

正しく入力したら、次へ(N) ボタンを押してください。

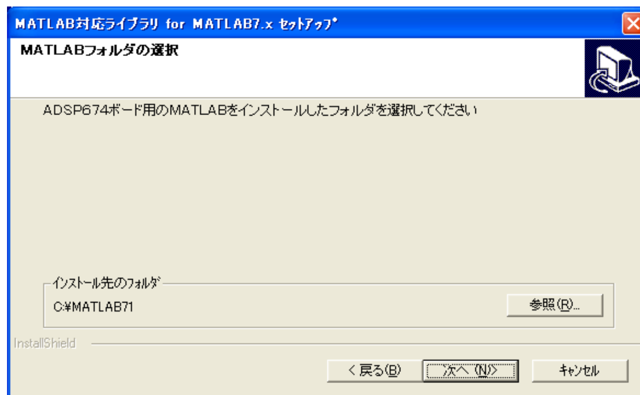
【７】MATLAB ﾊﾞｰｼﾞｮﾝの確認



インストールされている MATLAB のﾊﾞｰｼﾞｮﾝを選択してください。

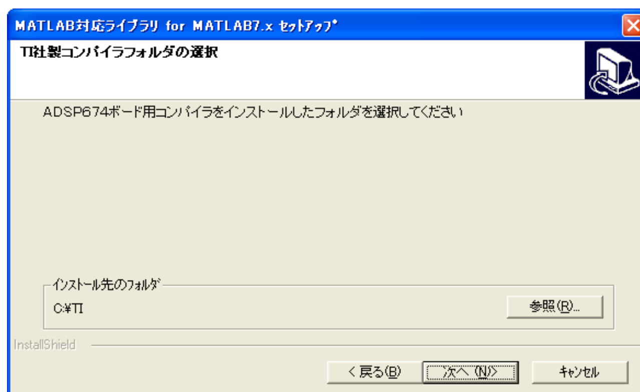
選択できましたら、 次へ(N) ボタンを押します。

【８】MATLAB のフォルダの選択



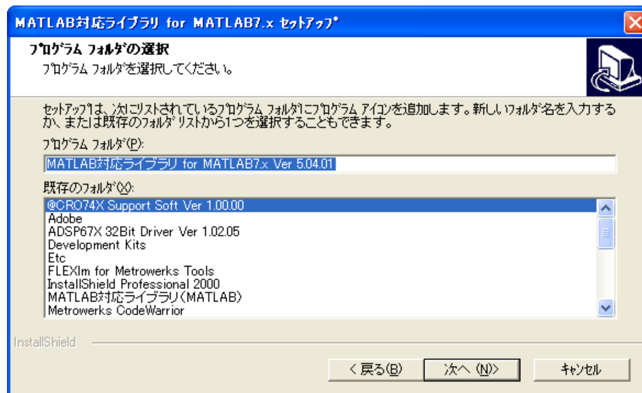
MATLAB がインストールされているディレクトリ（フォルダ）を指定します。
MATLAB をインストールしたディレクトリ（フォルダ）が表示されているディレクトリと異なる場合は、参照(R) ボタンを押しそのディレクトリ名を設定してください。
MATLAB のディレクトリが正しければ 次へ(N) ボタンを押します。

【９】Ｃコンパイラのフォルダの選択



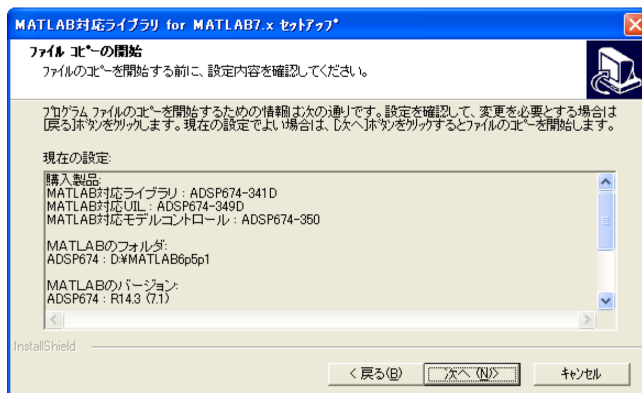
続いて、DSP の開発環境（TI 社製 C コンパイラ）をインストールしたフォルダを指定します。
コンパイラをインストールしたフォルダが表示と異なる場合は、参照(R) ボタンを押し、そのフォルダ名を設定してください。
フォルダ名に空白が含まれていると、モデルビルド時にエラーとなります。空白が含まれているフォルダ名へインストールされている場合は、このインストールを中止し、コンパイラのインストールをやり直して（アンインストールし、インストール実行）から、再度インストールしてください。
DSP の開発環境のフォルダが正しければ、次へ(N) ボタンを押してください。

【１０】プログラムフォルダの選択



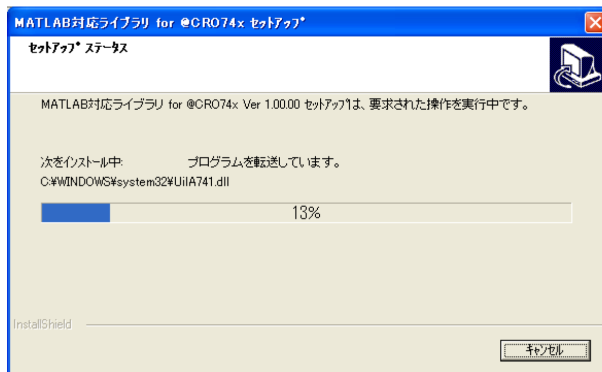
次に、ショートカットを追加するプログラムフォルダの選択をします。
表示されているフォルダ以外のフォルダにショートカットを追加する場合は、そのフォルダ名を選択又は入力します。
内容を確認したら、次へ(N) ボタンを押してください。

【１１】ファイルコピーの開始



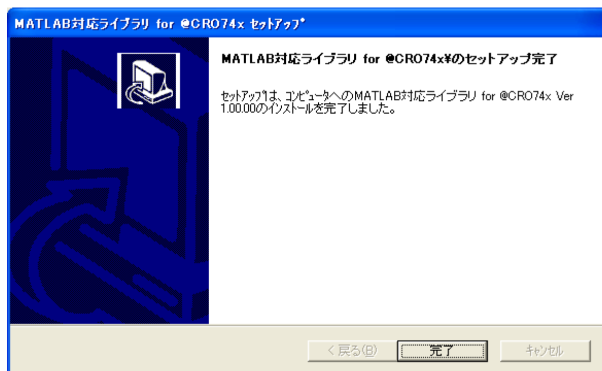
ファイルコピーの開始画面が表示されます。表示内容を確認してください。
内容が誤っているようであれば、戻る(B) ボタンで元に戻って再設定できます。
内容を確認したら、次へ(N) ボタンを押してください。

【１２】セットアップの実行



セットアップステータスが表示され、ファイルのコピーが自動的に行われます。
画面の表示及びプログレスバーがインストール作業の進行と共に変化します。

【１３】ウィザードの完了



最後に、“ＭＡＴＬＡＢ対応ライブラリ for ＭＡＴＬＡＢ　７．ｘのセットアップの完了“、が表示されます。
完了ボタンを押してください。これでインストールは完了です。

４．異なる種類のＤＳＰの同時利用

ＭＡＴＬＡＢ対応ライブラリ／ＭＡＴＬＡＢ対応ＵＩＬ／モデルコントロールは、以下の表に示す種類のＤＳＰボードに対応しています。

ＤＳＰボード	MATLAB 対応ＬＩＢ	MATLAB 対応ＵＩＬ	モデルコントロール
ADSP324-00A	ADSP324-41D	ADSP324-49D	ADSP324-50
ADSP674-00/00H	ADSP674-341D	ADSP674-349D	ADSP674-350

同じ種類のＤＳＰは１つのＭＬＬＩＢ／ＭＬＵＩＬ／モデルコントロールで動作させる事が出来ます。
２種類以上の異なる種類のＤＳＰを使用するは、それぞれのＤＳＰ用のソフトウェアをインストールする必要があります。その場合の注意点について述べます。

複数の製品をインストールする場合以下の注意点が必要です。

◎ ＭＡＴＬＡＢを複数それぞれのＤＳＰ用にインストールしてください。１つのＭＡＴＬＡＢ上に２種類以上のＤＳＰ用の環境をインストールする事は出来ません。

◎ 作業フォルダーを区別してください。同じフォルダーで異なるＤＳＰ用のコードを作成する事は出来ません。

◎ ブロック線図の共用は可能です。同じ版のＭＡＴＬＡＢ間であれば、例えば ADSP324-00A 用に作成したブロック線図を、ADSP674-00 用のフォルダーにコピーし、そこで ADSP674-00 用のコードを作成する事は可能です。

５．ＡＤＳＰ３２４－００Ａを使用する際の注意事項

ＡＤＳＰ３２４－００Ａの場合にリアルタイムモニタ、ＵＩＬ、モデルコントロールで使用するＤＬＬ（ダイナミック・リンク・ライブラリ）はＵＳＢ接続との互換性を重視し、ＩＳＡ接続でのパソコンーＤＳＰ間の通信速度が低下しています。ＤＳＰの動作速度には関係ありませんが、過去にＩＳＡ接続で使用したパソコン側（ＲＴＷでのモニタ、ＵＩＬを使用したＶＢプログラムなど）の動作速度が間に合わない場合はＩＳＡ専用ＤＬＬ（前バージョンのもの）を使用して下さい。今後、新規に作成する場合は共用ＤＬＬを使用して下さい。切り替え方法は スタート～プログラム～ＭＡＴＬＡＢ対応ライブラリ～ＤＬＬ切替～「ＩＳＡ専用ＤＬＬへ切り替え」 をクリックして下さい。共用ＤＬＬへ戻す場合は 上記同様～「共用ＤＬＬへ切り替え（標準)」 をクリックして下さい。

ＵＳＢ接続にするには別売のＵＳＢ接続ボード ADSP324-146 が必要です。
この作業は WindowsNT/2000/XP の場合は Administrator 権限で行って下さい
この問題はスタンドアローン時には関係ありません。
ＡＤＳＰ６７４－００/００Ｈは当初より互換性があります。

接続に関する詳細は ADSP324-00A スタートアップガイド、ADSP674-00 スタートアップガイドを参照して下さい。
最新版のマニュアル、サポートソフトはホームページより無料でダウンロードできます。
中部電機ホームページＵＲＬ(http://www.chubu-el.co.jp)

６．ＭＡＴＬＡＢ対応ライブラリ実行環境について

コンパイラのバージョンによって、コンパイラの実行に必要な以下の環境変数の設定が動的に実行するようになっています。この設定をする実行ファイルは”コンパイラのインストールフォルダ¥dosrun.bat”にあります。ＭＡＴＬＡＢを起動するにあたって必要に応じて先にこのＢＡＴファイルを実行し環境設定をしてください。

サンプル)

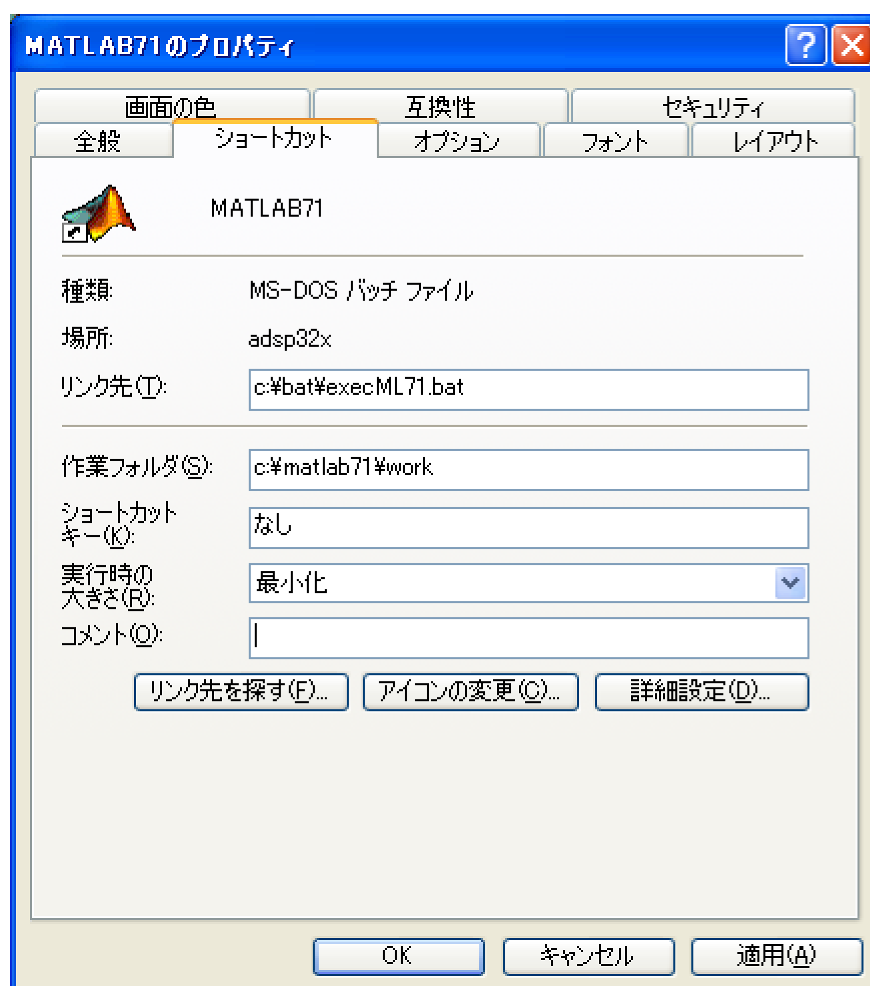
① 起動用ＢＡＴファイルを作成する。

ＢＡＴファイル内容(コンパイラが c:¥CCStudio_v3.1,ＭＡＴＬＡＢが c:¥matlab71 にインストールされている時)・・・

```
call c:¥CCStudio_v3.1¥dosrun.bat
call c:¥matlab71¥bin¥win32¥matlab.exe
exit
```

② ショートカットにより①で作成したＢＡＴファイルを起動する。

ＢＡＴファイルが　c:¥bat¥execML71.bat の場合



【　改　訂　履　歴　】

改訂番号	改訂日付	改　訂　内　容
初版	2006.09.14	初版
第２版	2006.10.24	MATLAB 実行環境について追記

MATLAB 対応ライブラリ
インストールガイド

中部電機株式会社
〒440-0004　　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　　FAX <0532>63-1081
URL：http://www.chubu-el.co.jp
E-mail：csg@chubu-el.co.jp

2006. 10　　第２版発行

ＡＤＳＰシリーズ

ＲＴモデル自動生成

ＭＡＴＬＡＢ対応ライブラリ

中部電機株式会社

目　次

１．概要

MATLAB のツールボックスのひとつ、Real-Time Workshop を用い、C コードを生成し、弊社製ＤＳＰ

ﾎﾞｰﾄﾞ（ADSP324-00A,ADSP674-00H/00）で実行することができます。また、本製品のリアルタイムモニタ（以下　RTMON）を使用すれば、ただちに弊社製ＤＳＰﾎﾞｰﾄﾞ上で実行出来、さらに、その結果をファイルに残すことができます。

　ここでは、SIMULINK 上でのシミュレーションのモデル作成は完了しているものとし、これを元にリアルタイムモデル（以下　ＲＴモデル）を生成する手順について説明します。

２．　使用上の注意事項

２．１．　バージョンアップ後のコード生成

　本製品のバージョンアップを行った場合、SIMULINK ブロック線図からＲＴモデルを作成する際に、一旦、モデルと同じフォルダにあるサブフォルダ　“ブロック線図名_grt_rtw”を削除してからコード生成を行ってください。その際 I/Odefine ブロックによる I/O ボードの設定もおこなってください。

２．２．　モデル名称の制約

　ＲＴモデルの名称（ブロック線図のファイル名）には、以下に示す制約があります。モデル名称作成の際には、充分注意をしてください。

- 名称の先頭は、英字で始めてください。
  - 良い例：　a3bc.mdl
  - 悪い例：　3abc.mdl　（理由＝数字で始まっている）
- 名称で使う文字は英数字と　＿　記号のみとしてください。
  - 良い例：a32_abc.mdl
  - 悪い例：a(#1).mdl　　（理由＝かっこ、#等の記号を使っている）
- 英字の大文字と小文字は同じものとして取り扱われます。

［注意］

　ADSP324-00A の場合モデル名称の長さによりビルド時に、下記の Warning ﾒｯｾｰｼﾞが表示されることがあります。（コンパイラの制約）このﾒｯｾｰｼﾞが表示された場合はモデル名称を短くしてください。

　Ex)Ver5.1 の場合モデル名称６文字以上は表示されます。

```
warning: Output filename, test11_data.asm, truncated for desired COFF format
```

２．３．ＤＳＰ用コンパイラー版番号

　使用出来るＤＳＰ用コンパイラーは以下の通りです。

［ADSP674-00H/00］

◎ Code Composer Studio Ver 1.0/2.1/2.2/3.0

［ADSP324-00A］

◎ CPL320FLT-MS 5.1 版(Code Composer Studio 対応版)

◎ CCS320FLT-MS 5.1 版

３．ＳＩＭＵＬＩＮＫブロックの制約

SIMULINK 標準のブロックライブラリの一部は、ＲＴモデルに於いて利用出来ないか、又は利用上の注意が必要です。この章ではこれらについて解説しています。

３.１． simulink Sources ～ Pulse Generator
３.２． simulink Sources ～ Signal Generator

Pulse Generator／Signal Generator は、ＤＳＰが取り扱う時刻変数の精度の関係で、実際の時間（又は SIMULINK で実行した仮想時刻）より、１ステップサイズ分、前後する場合があります。
ＤＳＰは変数を全て単精度実数（32Bit＝8E24）で取り扱います。この精度は、１０進数換算で有効桁数７桁程度です。例えば 0.003 はＤＳＰでは、0.00299999979324 となります。この誤差は避けられません。

３.３． simulink Sources ～ From File

RTW にてコード生成をした時点のファイルの内容が、生成される C ソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、ＲＴモデルの実行時点のファイル内容とは無関係となります。
ファイル内容を変更し､その結果をＲＴモデルに反映させるには､再度 RTW でコード生成が必要です。
データが実行コードに組み込まれる関係上、その長さは、ＤＳＰのメモリ容量を超える事は出来ません。

３.４． simulink Sources ～ From WorkSpace

RTW にてコード生成をした時点のワークスペース変数の内容が、生成される C ソースコードにデータテーブルの形で組み込まれます。組み込まれたデータはその状態で実行形式コード内に固定され、ＲＴモデルの実行時点のワークスペース内容とは無関係となります。
ワークスペース内容を変更し、その結果をＲＴモデルに反映させるには、再度 RTW でコード生成が必要です。
データが実行コードに組み込まれる関係上、その長さは、ＤＳＰのメモリ容量を超える事は出来ません。

３.５． simulink Sinks ～

Sinks グループの Stop Simulation を除く他のブロック（Scope 等）は、RTW では利用できません。(MATLAB4.2c では Stop Simulation に加え Scope と ToWorkSpace は利用可能)。
Scope は RTMON の変数モニター機構を使って代用が可能です。具体的には、Scope へ接続されている信号の出力元のブロック名称を指標して変数表示を行います。

３.６． simulink Nonlinear ～ MATLAB Fcn

RTW では利用出来ません。sin 、 cos 等、一部の関数は、Fcn ブロックで代用が可能です。

３.７． simulink Nonlinear ～ Algebratic Constraint

RTW では利用出来ません。

３.８.　simulink Nonlinear 〜 Fcn

利用は可能ですが、式の定義に注意が必要です。

（1）C 言語の標準ランタイムライブラリでサポートされていない関数は利用出来ません。利用出来る関数／演算子は、以下の通りです。

利用出来る関数	Ｃ言語展開
sin	sin
cos	cos
tan	tan
asin	asin
acos	acos
atan	atan
atan2	atan2
sinh	sinh
cosh	cosh
tanh	tanh
pow	pow
exp	exp
log	log
log10	log10
sqrt	sqrt
floor	floor
ceil	ceil
abs	fabs
mod	fmod
rem	fmod
sgn	sgn　（注１）
hypot	hypot

利用出来る演算子	Ｃ言語展開
＞	＞
＜	＜
＝＝	＝＝
！＝	！＝
＞=	＞=
＜=	＜=
&&	&&
｜｜	｜｜
（ ）	（ ）
＋	＋
−	−
／	／
＊	＊
！	！

注１：標準ランタイムには含まれませんが、ＲＴＷ用拡張ライブラリが対応しています。

（2）式中に定数を記述する場合で、分数の分子・分母に整数を置く場合は、整数であっても少数形式で記述してください。例えば、1/2 を記述する場合は、
　1／2.0 又は、1.0／2、又は、1.0／2.0 等としてください。
RTW ではFcn ブロックのExpresion に記述された式を、殆どそのままＣ言語ソースに展開します。よって単に 1／2 と記述すると、C 言語の文法規約上、整数計算が行われ、1/2 は少数以下切り捨られ 本来0.5 が欲しい所が、＝0 となってしまい、計算結果に重大な誤りを招きます。

３.９.　Simulink Extras 〜 Additional Discrete 〜 Discrete Zero-Pole

Discrete Zero-Pole (with initial states)は、Simulink の純正ブロック自体が Bug を孕んでいます。以下の手順で対策してください。

（１）MATLAB Command wondiw から Simulink を開く。
（２）Blocksets & Toolboxes をマウス左ダブルクリックし、
　　Library: Blocksets_and_Toolboxes を開く。
（３）同 Simulink Extras をマウス左ダブルクリックし、Library: Simulink_Extras を開く。
（４）同 Additional Discrete ををマウス左ダブルクリックし、
　　Library: Simulink_Extras／Additional Discrete を開く
（５）同 Discrete Zero-Pole (with initial states) をマウスで選択する。
（６）メニューから Edit 〜 Unlock Library を選択する。
（７）さらに、Edit 〜 Look Under Mask を選択し、Library: Simulink_Extras／Additional Discrete ／Discrete Zero-Pole (with initial states) を開く。
（８）同、Discrete State Space をダブルクリックし、Discrete State Space ダイアログを開く。
（９）エディットボックス D:の内容が Dd となっているので、これを D に変更する。
（１０）同ダイアログの Close ボタンを押す。
（１１）先の（７）で開いたウィンドウを閉じる。
（１２）先の（４）で開いたウィンドウを閉じる。
（１３）（３）で開いた Library: Simulink_Extras のメニューから、File 〜 Save で変更結果を保存する。

３.１０.　　Simulink Extras 〜 Flip Flops 〜 D Flip Flop

D 入力=1 の状態で、CLK の立ち上がりと、！CLR＝0 が同時に発生する状況では、SIMULINK の実行結果と ＲＴモデルの実行結果に相違が現れます。
SIMULINK では CLK が優先され Q=１となり、ＲＴモデルでは、!CLR が優先され Q=0 となります。

４． MATLAB での作成手順

４．１． 作成に必要なファイル

ＲＴモデルの作成に必要なファイルは、ＲＴモデルを記述したブロック線図です。ブロック線図名をEXAMPLEとするとブロック線図をファイルに保存するとEXSAMPLE.MDLというファイルとして保存されます。

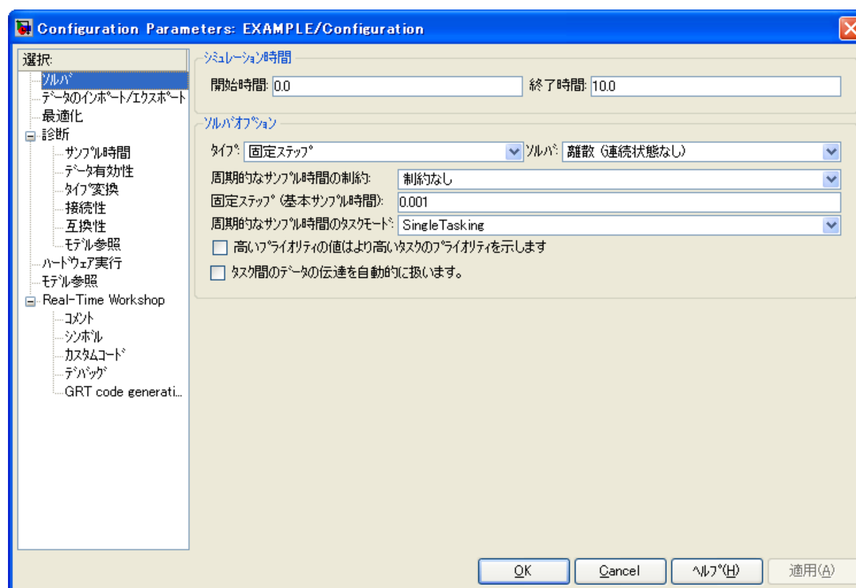
４．２． オプションの設定

モデルのブロック線図を開き、ブロック線図のメニューから ﾂｰﾙ(T) の Real-TimeWorkshop->ｵﾌﾟｼｮﾝ(O)をマウス左ボタンでクリックし、選択します。すると、Simulation Parameter ダイアログウィンドウが開きます。リアルタイムモデルのＤＳＰ内での実行条件（実行時可変な一部パラメータを除く）の設定はここで行います。

Simulation parameters: ダイアログの ソルバ タブをマウス左クリックします。
ｿﾙﾊﾞｵﾌﾟｼｮﾝ を固定ｽﾃｯﾌﾟ に変更します。これはリアルタイムモデルでは一定周期での計算しか行えない事を意味しています。
続いて、積分則を選択します。モデルが連続系の状態変数を持たない場合は 、離散（連続状態無し）を選択、それ以外の場合はリアルタイムモデルでデフォルトとして利用したいと思う積分則を選択します。

さらにすぐ下の 固定ステップサイズ：に具体的なステップサイズ値を設定します。単位は秒です。図４-1 ではステップサイズが１mSecに設定されています。ここで設定した積分則とステップサイズは、RTMONでその設定をデフォルトとして取り出す事が出来ます。
タスクモードは、Single Tasking を選択してください。

図 ４-1 積分アルゴリズム等設定後

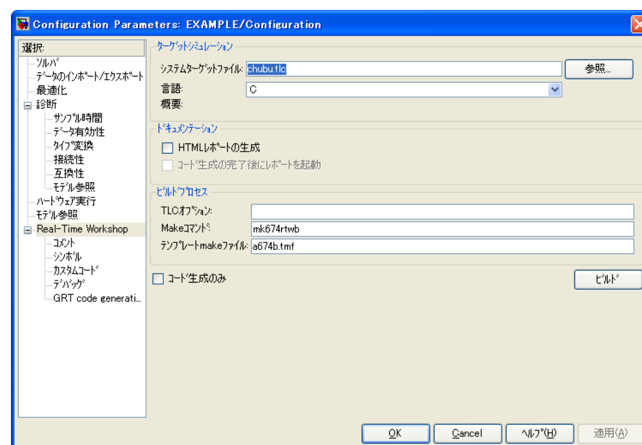


次に、Real-Time Workshop タブをマウス左クリックし、以下を確認します。

	ADSP674-00H/00 の場合	ADSP324-00A の場合
ｼｽﾃﾑﾀｰｹﾞｯﾄﾌｧｲﾙ:	chubu.tlc	chubu.tlc
Make ｺﾏﾝﾄﾞ:	mk674rtwb(mk674rtw)	mk32xrtw
ﾃﾝﾌﾟﾚｰﾄ Make ﾌｧｲﾙ:	a674b.tmf(a674.tmf)	a32x.tmf

図 4-2 は、これらの設定が終わった例です。
通常、新規に作成したばかりのブロック線図にはこの設定がデフォルトとして設定されています。この設定は、リアルタイムモデル用のタイマードリブンモデルを生成する場合の設定です。それ以外の実行形式モデルの生成方法は ４．３． 実行形式の選択 (8p) を参照してください。

図 4-2 ＲＴＷオプション（設定後）



設定を変更した場合は適用 ボタンを押して設定を確定してください。

出来ればこの段階で、上記設定を保存する意味で、一旦モデルを保存する事をお勧めします。ブロック線図メニューから ﾌｧｲﾙ(F) → 名前を付けて保存(A) 又は、ファイル(F) →上書き保存(S) を実行します。

コード生成は、同ダイアログの ビルド ボタンを押してください。

MATLAB コマンドウィンドウに、 ### で始まるメッセージが何行か表示されます。(一部###以外で始まる行が含まれる場合がありますが、これは異常ではありません。)
これらの行は全て RTW 関連のメッセージです。コード生成に続きコンパイル・リンク作業が行われ、以下が表示されればコード生成は完了です。
Successful completion of Real-Time Workshop build procedure for model: XXXXX
(注 XXXXX は作成するモデルの名称です。)

ブロック線図作成後初めてコード生成をする場合と２回目以後の再度コード生成する場合とでは表示される行数が異なります。これは、ブロック線図変更に無関係のソースファイルは２度目のコンパイル作業を省略し、必要最小限のソースファイルのみコンパイルしている為です。

注) 新規以外で作成されたモデルをコード化する場合、弊社ホームページの Q&A「MATLAB 対応ライブラリでよくある Q&A」を参照してください。

４．３．　実行形式の選択

RTW によるＤＳＰ用リアルタイムモデルは、その実行方法とモデル計算のきっかけとなるイベントが選択可能です。
実行方法は、RTMON 又は UIL による方法“パラメータ可変モデル”と、自動的に実行開始する“スタンドアロンモデル”が可能です。
パラメータ可変モデル は、ステップサイズ、積分アルゴリズム、ゲインブロック等のパラメータ等が実行時に可変で、データロギング、リアルタイムなモニター等が行えます。未完成のモデルを評価したり、パラメータを種々変更してテストする場合等に有用です。
スタンドアロンモデル は、パラメータ等はRTWにてコード化される時点の設定で固定され、実行時には可変出来ない代わりに、プログラムをＤＳＰにロードするとただちに実行が開始されます。これを応用し、ROM化（オプションハードウェアが必要）する事により、電源投入と同時に自動的に動作を開始するスタンドアロンモデルを作る事ができます。

表　４．３-1　実行方法別モデル機能一覧

名称	自動実行	ROM 化	ロギング	ﾊﾟﾗﾒｰﾀ変更	モニタ
パラメータ可変	×	×	○	○	○
スタンドアロン	○	○	×	×	×

モデル計算のきっかけとなるイベントは、タイマーイベント、外部割り込みイベントが選択出来ます。
タイマーイベント は、最も一般的で、ＤＳＰのタイマーを用いて一定周期のタイマーイベントを発生させ、これにより、ブロック線図の計算を行います。この周期がRTWによるコード生成時の Step size に相当します。
外部割り込みイベント は、ブロック線図の計算を外部からの割り込みイベントをきっかけとして行います。これにより、外部CPUや外部ハードウェアに同期したモデル計算が可能となります。SIMULINK の可変ステップサイズと似た動作をしますが、SIMULINKの場合は時間刻みの増減の基準が状態変数の微係数の大小であるのに対し、イベントドリブンは時間刻みが“外部割り込みイベントの発生”に完全に依存します。割込みイベント処理は、rt5event.c（MATLABｲﾝｽﾄｰﾙﾌｫﾙﾀﾞ¥rtw¥c¥acro74x¥rt741）のコーディングが必要となります。
実行方法とイベントは、コード生成の際に RTW Option ダイアログへの設定で切り替えます。

表　４．３-2　実行形式・イベント切り替え一覧

イベント	実行方法	上段：Template make file 下段：Make command
タイマー	パラメータ可変	a674.tmf mk674rtw
	スタンドアロン	a674.tmf mk674rtw　ALONE=ON
外部割り込み	パラメータ可変	a674.tmf mk674rtw　EVENTD=ON
	スタンドアロン	a674.tmf mk674rtw　EVENTD=ON　ALONE=ON

☆　Large モデル時 a674.tmf → a674b.tmf
　　mk674rtw → mk674rtwb を使用してください。
　デフォルトは、Matlab 対応ライブラリ Ver5.2.0 以降は Large モデルです。
☆　ADSP324-00A 時 a674.tmf → a32x.tmf
　　mk674rtw → mk32xrtw を使用してください。

４.４.　作成されるファイル

Real-Time Workshopで作成作業を行うと、MATLABのカレントディレクトリに、”モデル名_grt_rtw” という名称のサブディレクトリが生成され、その中に、表 ４.４-1に示すファイルが作成されます。作成作業を行う前のカレントフォルダーに、これらと同名のファイルがあると、上書きにより以前の内容は失われてしまいます。カレントフォルダーにユーザのファイルを置く場合には注意が必要です。作業は順を追って自動的に行なわれます。最終的に実行に必要となるファイルは、EXAMPLE.out です。

表 ４.４-1　自動生成されるファイル一覧表

番号	生成ファイル名	作成主	備考
1	EXAMPLE.C EXAMPLE_DATA.C EXAMPLE.H EXAMPLE_TYPES.H EXAMPLE_PRIVATE.H EXAMPLE.RTW Rt_nonfinite.h Rt_nonfinite.c Rtwtypes.h Adspiod.h	Real-Time Workshop iodef.exe	主となるソースファイル
2	EXAMPLE.MK EXAMPLE.LK EXAMPLE.LCF EXAMPLE.OPT EXAMPLE.ITB	Buil command	実行ファイルの作成に 必要なファイル
3	*.obj	コンパイラ	オブジェクトファイル
4	EXAMPLE.elf EXAMPLE.map	リンカ	実行形式ファイル 番地情報ファイル

注:EXAMPLE は、ユーザが命名したモデルの名称です。Real-TimeWorkshop が生成する C/H ファイルはバージョンにより多少異なります。

４.５.　実行手順（RTMON の場合）

RTMON の場合は、Windows のプログラムマネージャで、RTMON を起動し、これによりロード・実行等を行います。この方法は画面上でパラメータや、起動停止等をマニュアリックに操作するので、自動起動等は行えませんが、最も柔軟性に富んだ方です。
詳細は、RTMON の章を参照してください。

４.６.　実行手順（自動実行の場合）

自動実行は、ＤＳＰ付属ソフトウェアのユーティリティを用いて ROM 化します。詳しくは 該当する ADSP324-00A,ADSP674-00H/00 ソフトウェア・ユーザーズマニュアルを参照してください。

５． ステップサイズについて

５．１． SIMULINK との関係

RTW によるＲＴモデルを生成する場合、実時間モデルでは、可変時間刻みは実現出来ません。実時間モデルでは、過去の時刻にさかのぼって計算をする事が出来ないからです。

可変時間刻み（Variable-Step）は、微係数を監視して、傾きが小さい場合は時間刻みを粗くして計算省略により時間を短縮すると共に、傾きが大きい場合は時間刻みを細かくして計算精度を維持する意図を併せ持っています。微係数の傾きが大きい場合は自動的に時間刻みが細かくなりますが、固定刻みの場合はこれが行われない為、誤差が増大する可能性があります。
従って、最終的に実時間モデルを作成する予定がある場合、SIMULINK での評価の段階で、固定刻み（Solver Type を Fixed-Step）にして評価する必要があります。

５．２． ステップサイズのデフォルト

RTMON では､RTW で生成したＲＴモデルのデフォルトステップサイズを取り出す機能を持っています。
この機能で取り出せる情報は、モデルの内容により、若干異なります。
この機能で取り出せる情報と、SIMULINK（RTW との関係）は、以下の表の様になっています。

モデルの内容	RTMON のﾃﾞﾌｫﾙﾄ
連続時間系のブロックのみ	RTW の Simulation paramet- er の Fixed-step
連続時間系の離散時間系ブロック	一番小さいｻﾝﾌﾟﾙﾀｲﾑ
離散時間系のブロックのみ	（同上）

６． 速度向上について（ADSP674-00H/00 の場合）

６．１． 実行速度の相違

ADSP674-00 では、使用している DSP チップの特性上、プログラムの配置により、実行速度が異なります。ボード上に、ＳＢＳＲＡＭ、ＳＤＲＡＭと、チップ内部にオンチップＲＡＭがあります。最も早いのはオンチップＲＡＭですが、チップ上に実装されているオンチップＲＡＭは容量に限りがあります。オンチップＲＡＭより外部のＳＢＳＲＡＭの方が容量が大きいので、大きなプログラムは外部に置きます。

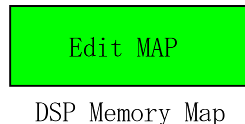
６．２． 高速化するには

本ソフトでは、ユーザが作るプログラム（ブロック線図）の大きさが判らない為、標準では全て外部のＳＢＳＲＡＭに実装します。この状態で、希望とする速度が満たせられれば良いのですが、満たせられない場合、プログラム等の配置をオンチップＲＡＭに変更します。
配置を変えるには以下の手順を踏みます。

１、ＭＡＰ変更ツールを使い、配置指定を変更します。
２、ＲＴＷでコード生成をします。
３、メモリーが不足する旨のエラーが出たら１へ戻り、内部に割り付けた領域を外部に戻します。
４、エラーが出なければ完了です。

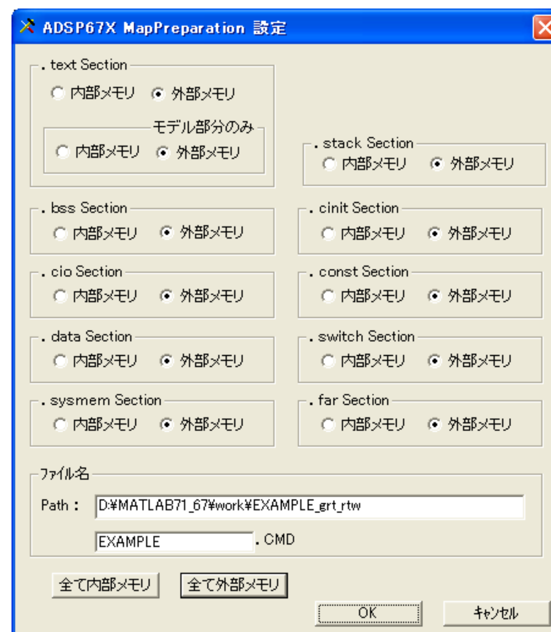
６．３． ＭＡＰ変更ツール

ＭＡＰ変更ツールは、Ｉ／Ｏブロックライブラリのブロック線図に有る、“ＭＡＰエディットボタン”（下図）を目的とするブロック線図に貼りつけ、これをダブルクリックします。



DSP Memory Map

次のような画面が開きます。



本ソフトではＤＳＰに実装されるユーザプログラムを、それを構成する要素（セクション）毎に分けて、そのセクション毎に配置する場所をチップの内部か外部か選択できる様になっています。
.text Section 等のグループは、個々のセクションを表しています。
ファイル名のグループは、これらの情報を保存するフォルダーとファイル名を表しています。この名称は、MAP 変更ツールをブロック線図から呼び出した場合は変更してはいけません。又、このツールにより自動生成されたファイル（上の例では untitled.cmd）は、直接変更してはいけません。

配置の変更は、各セクションのグループのラジオボタンで、内部メモリか外部メモリかを選択し、

ＯＫ ボタンを押します。設定された内容が書き込まれます。

変更後の内容は次回このツール起動時に表示されます。現在の配置を確認する目的でこのツールを起

動した場合は、 ｷｬﾝｾﾙ ボタンを押して下さい。

設定（又は表示）が終わったら MAP 変更ツールを終了してください。MAP 変更ツールを起動したままでは、MATLAB/SIMULINK の操作は出来ません。

各要素のおよその意味は以下の表の通りです。

＜セクション名と意味＞

番号	セクション名	意味	効果	割付先
1	.text	プログラムの命令コード	◎	OCPMEM
2	.bss	大域変数、static な局所変数	◎	OCDMEM
3	.cio			
4	.data	ASM 用データ領域		
5	.sysmem	ヒープ領域	◎	
6	.stack	システムスタック	◎	
7	.cinit	C 言語用初期化データ		
8	.const	大域定数、static な局所定数データ		
9	.switch	分岐テーブル	◎	
１０	.far	Far 参照用アドレステーブル	◎	

実行時間に大きく影響を及ぼすと思われるものは表の効果の覧に◎を付けてあります。

６．４． メモリー不足エラー例

メモリーが不足すると、RTW によるコード生成の最後の段階で、MATALB Command Window にエラーメッセージが表示されます。

```
>> cannot allocate .text in OCPMEM (page 0)
>> cannot allocate .bss in OCDMEM (page 0)
>> cannot allocate .cinit in OCDMEM (page 0)
>> cannot allocate .sysmem in OCDMEM (page 0)
>> cannot allocate .far in OCDMEM (page 0)
>> errors in input - untitled.out not built
NMAKE : fatal error U1077: 'e:¥ti¥C6000¥CGTOOLS¥bin¥lnk6x.exe' : ﾘﾀｰﾝ ｺｰﾄﾞ '0x1'
Stop.
```

>>cannot allcate の後ろに割り付けできなかったセクション名(.text 等)とその割付先（OCDMEM 等）が表示されています。
エラーが出た場合の対策は、表示されたセクション、又は、同じ割付先（OCPMEM／OCDMEM）の別のセクションを外部メモリーに割り当てます。

７． 速度向上について（ADSP324-00A の場合）

７．１． 高速化するには

ＲＴモデルを動作させて処理時間が希望するステップサイズより長かった場合の対策は、次の２つのうちの何れかです。

１．ステップサイズを長くする
２．処理時間をステップサイズ以内に終えられる様に短くする。

上記１は別の問題を孕みますので、ここでは２の方法について説明します。
処理時間を短くする方法は次の２つの何れかです。

１．処理の内容を低減する。
２．処理を最適化する。

処理内容の低減は、ブロック線図の基本的な設計変更が必要ですし必ずしも可能とは限りませんから、これも別の問題を孕みますのでここでは検討外として、処理の最適化について説明します。
処理の最適化は、ブロック線図の段階で行うものと、Ｃ言語のコードが出来てから行うものとあります。
ブロック線図で行える最適化には以下のものがあります。

◎ＡＤ変換器等Ｉ／Ｏブロックを使っている場合で、１チャンネルのブロックを３個並べている場合、これを複数チャンネルのブロック１個に置換え、チャンネル数を３と設定します。これの出力を Demux ブロックを使って３本のスカラーに変換します。これにより I/O に費やす時間が低減されます。
◎ＡＤ変換器等Ｉ／Ｏブロックを使っている場合で、複数チャンネルのブロックを使っていても、チャンネル数が１の場合これを１チャンネルのブロックに置換えます。これにより I/O に費やす時間が低減されます。
◎ゲインブロックを複数シリアルに接続している場合、これらを１個のゲインブロックに置換えます。

C コードが出来てから行う最適化で最も簡単なものは、コンパイラによるコード生成の最適化です。本ソフトでは、確実に動作する事を優先的に設計されていますので、最適化レベルはデフォルトに設定されています。この最適化レベルを上げる事により高速化が見込める場合があります。しかし、最適化は時として意図しない結果をもたらす場合（正しく計算されない場合）が有る為、やみくもに最適化するのは危険です。従って最適化のレベルの変更は以下の手順で行います。

１．次のファイルをコピーし同じフォルダーに別の名称で複製します。
　　元のファイル　　　C:¥Matlabr12¥rtw¥c¥adsp32x¥a32x.tmf
　　複製後のファイル　C:¥Matlabr12¥rtw¥c¥adsp32x¥a32x2.tmf
２．コード生成の際の“テンプレート Make ファイル”の設定を以下の様に変更します。
　　変更前　　a32x.tmf
　　変更後　　a32x2.tmf
３．複製したファイルの内容のうち以下の様な @$(CC) で始まる行を探し、その直後にコンパイラの最適化レベルをコントロールするスイッチを記入します。
　　変更前　　@$(CC) $(RTI)¥$*.c
　　変更後　　@$(CC) –o3 $(RTI)¥$*.c
　スイッチは –o0 、–o1 、–o2 、–o3 、（マイナス、小文字の 0 オー、数字、の組み合わせ）で、数字が大きい程最適化が強化されます。
　対象となる行は複数有りますが、全部を一度に変更せず、一部分ずつ変更しては動作確認する、という作業を繰り返します。
４．オブジェクトファイルを削除します。ブロック線図が untitled の場合同じフォルダーの untitled_grt_rtw¥*.obj を全て削除します。
５．実行コードを作成し、正しく動作する事を確認します。もし希望通り動作しない場合は先に行

った最適化のレベルを下げるか又は最適化のスイッチを外してください。

８． ＦＡＱ（よくある質問～ADSP324-00A の場合）

８．１． コード生成時のエラー（１）cbios324.lib

コード生成時に発生する以下のエラーメッセージ（MATLAB Command Window）の原因及び対策は次の通りです。

【メッセージ】
>> untitled.lnk, line 29: can't find input file cbios324.lib
NMAKE : fatal error U1077: 'C:¥C30¥tic51¥BIN¥lnk30.exe' : ﾘﾀｰﾝ ｺｰﾄﾞ '0xff'
Stop.

【原因】
コード生成に必要なライブラリ Cbios324.lib が欠如しているか､正しく参照出来る位置に格納されていません。

【対策】
ＤＳＰボードに付属している Support Software に含まれている下記ファイルを、環境変数 C_DIR が指標しているフォルダーに格納してください。
Cbios324.lib、Cbios324.h

８．２． コード生成時のエラー（２）rtsb.lib

コード生成時に発生する以下のエラーメッセージ（MATLAB Command Window）の原因及び対策は次の通りです。

【メッセージ】
>> exe.cmd, line 3: can't find input file rtsb.lib
NMAKE : fatal error U1077: 'C:¥C30¥tic51¥BIN¥lnk30.exe' : ﾘﾀｰﾝ ｺｰﾄﾞ '0xff'
Stop.

【原因】
コード生成に必要なライブラリ rtsb.lib が欠如しているか､正しく参照出来る位置に格納されていません。

【対策】
ＤＳＰボードに付属しているスタートアップガイドを参照のうえ、ライブラリ rtsb.lib の作成を行い、環境変数 C_DIR が指標しているフォルダーに格納してください。

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2005.09.19	初版

ＡＣＲＯ７４１－４４１
ＭＡＴＬＡＢ対応ライブラリ
ＲＴモデル自動生成

中部電機株式会社

〒440-0004　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006. 9　第１版発行

ＡＤＳＰシリーズ

Ｉ／Ｏブロック取扱説明書

ＭＡＴＬＡＢ対応ライブラリ

中部電機株式会社

目　　次

１． ハードウェアのセットアップ

１．１． I/0 ボードアドレスの設定

DSPﾎﾞｰﾄﾞはI/0ボードの種類を問わず最大４台のI/0ボードを接続できるように設計されています。本製品はそれに対応し、合計４台までの各種I/0ボードを接続した場合を想定しています。各I/0ボードはアドレスで識別しますので、I/0アドレスの設定を適切に行なわなければいけません。

まず、各I/0ボードに個々のボードを識別する為の番号を、その種類毎に１台目から割り付ける事とします。例えば４種類のボードを各１台ずつ接続したとすると、何れも１台目となります。又、同じ種類のボードを複数接続する場合は、ボードの種類別に番号を飛ばさないように１台目から順に割り付けます。こうして割り付けた各ボードに対して 表 １．１－１　ボードアドレス設定表のボードアドレスを設定して下さい。ボードアドレスの具体的設定方法は各I/0ボードのハードウェア・ユーザーズ・マニュアルを参照して下さい。

表 １．１－１　ボードアドレス設定表

	Ｉ／Ｏボードの種類			
	ADSP324-03	ADSP324-06	ADSP324-11	ADPS324-13
１台目	900000h	900080h	900100h	900180h
２台目	900010h	900090h	900110h	900190h
３台目	900020h	9000A0h	900120h	9001A0h
４台目	900030h	9000B0h	900130h	9001B0h

	TM32DA	ADSP324-141	ADSP324-143	ADPS324-145
１台目	900200h	900280h	908000h	00901000h
２台目	900220h	9002A0h	90C000h	00901200h
３台目	900240h	9002C0h	910000h	00901400h
４台目	900260h	9002E0h	914000h	00901600h

２． MATLAB　I/O ブロックライブラリ

２．１．　概要

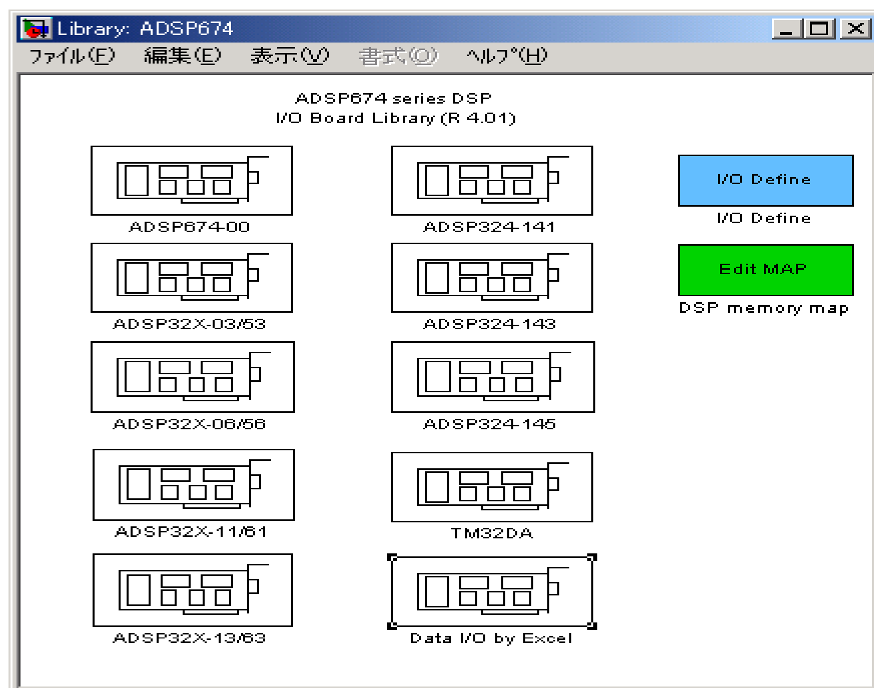
本製品のブロックライブラリは、SIMULINK の標準ブロックや、オプションのブロックセットと同様に、モデルに貼り付けて使用します。

こうしたモデルを Real-Time Workshop でコード化し、ADSP674 上で実行する際にこれらブロックにより、外部 I/O とデータ交換が出来ます。

２．２．　開き方

ブロックライブラリは、MATLAB コマンドウィンドウに次のコマンドを入力します。
>> ADSP674　（ADSP674-00H/00 の場合）
>> ADSP32x　（ADSP324-00A の場合）
すると、ＡＤＳＰ６７４－００Ｈ/００の場合は図 ２．２-１　ＡＤＳＰ６７４ウィンドウ　に示すブロックライブラリが開きます。ＡＤＳＰ３２４－００Ａの場合はＡＤＳＰ３２４用のウィンドウが開きます。

図 ２．２-１　ＡＤＳＰ６７４ウィンドウ



このウィンドウには、現在標準で供給されているＤＳＰ用の I/O ボード及び該当ＤＳＰ本体の I/O に関するデバイスドライバをボード毎に分類して収納してあります。
各 I/O ボードの型番を記したアイコン（ADSP324-03etc）が登録されています。これらのアイコンをダブルクリックすることにより、I/O ボードに対応したサブブロックのウィンドウが開きます。また、各モデル毎に I/O Define ブロックが必要です。このブロックを１モデルに１ブロック使用してください。内容は、実装する I/O ボードの枚数を設定するブロックです。このブロックの内容により I/O をアクセス制御します。

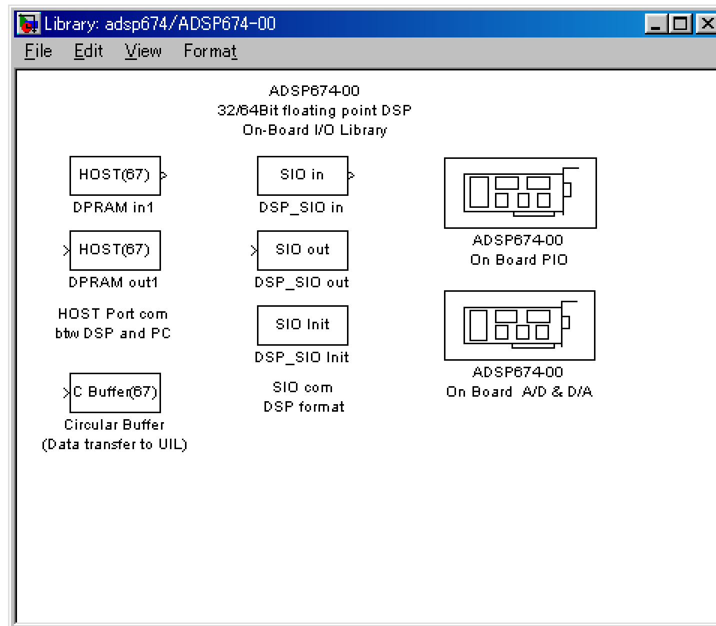
２．３．ADSP674 デバイスドライバブロック（ADSP674-00H/00 用）

２．３．１．　概要

図　２．３－１　ADSP674 ウィンドウ　(ADSP674 On Board I/O)を開いた所を示します。

図　２．３－１　ADSP674 ウィンドウ

この中には ADSP674 上の I/O（DPRAM 等含む）とのインターフェースを取るためのデバイスドライバブロックが登録されています。



HOST(67)	ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストとの間で非同期にデータ交換をする為のブロックです。
C Buffer(67)	ＤＳＰのサーキュラーバッファ機構を使った高速データバッファリングによるホストへのデータ送信を行うブロックです。高速に処理させたいブロック線図で使用する事により、データをバッファリングする為の負担を低減させることが出来ます。バッファリングしたデータは、ＵＩＬを用いたユーザアプリケーションへデータで取り出す事が出来ます。
SIO in/out/init	ＤＳＰのＳＩＯを使ったシリアル送受信を行うブロックです。ＤＳＰボード間での通信に使用します。

ADSP674-00 on Board PIO

ＤＳＰボード上のＰＩＯをアクセスするブロックを収納してある ADSP674-00 On Bard PIO ウィンドウを開きます。

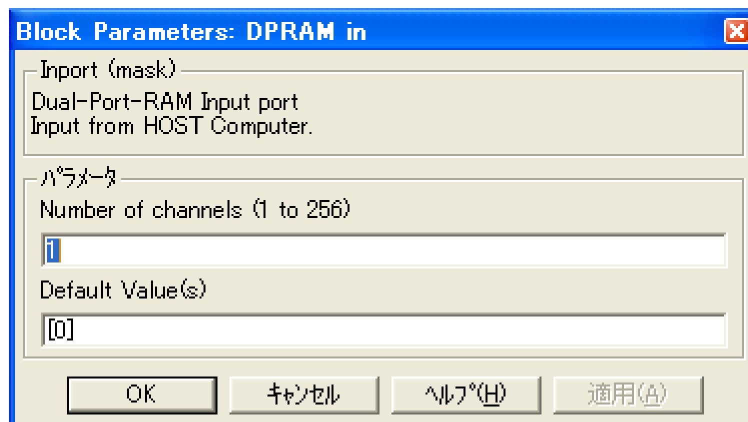
ADSP674-00 on Board AD & DA

ＤＳＰボード上のアナログＩＯ（ＡＤ及びＤＡ）をアクセスするブロックを収納してある ADSP674-00 On Bard AD & DA ウィンドウを開きます。

２．３．２．　ホスト DP 入力デバイスドライバブロック

ホストDP入力デバイスドライバブロックは、ADSP674 とホストとの間にあるDPRAMを使い、ホスト－ADSP674 間で非同期にデータ交換を行う為のブロックです。主に、UILを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、２．３．４ホストDP出力デバイスの概念 を参照してください。

図　２．３－２ HOST DPRAM in



- Number of Channels (1 to 256)
  ホストから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Value(s)
  ホストが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数　(Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックが通信に使用する領域は、ADSP674 から見たメモリマップの、3800H 番地から連続する 256 ワードを使っています。3800H 番地がベクトルの最初の要素です。データ形式は、ADSP674 の実数フォーマットです。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストと ADSP674 のアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

このブロックは１つのブロック線図に１個しか配置できません。

２．３．３．　ホスト DP 出力デバイスドライバブロック

ホストDP出力デバイスドライバブロックは、ADSP674 とホストとの間にあるDPRAMを使い、ホストーADSP674 間で非同期にデータ交換を行う為のブロックです。主に、UILを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、、２．３．４ホストDP出力デバイスの概念 を参照してください。

図 ２．３－３ HOST DPRAM out



- Number of Channels (1 to 256)
  ホストへ送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルにまとめてからこのブロックに入力します。

このブロックが通信に使用する領域は、ADSP674 から見たメモリマップの、3C00H 番地から連続する２５６ワードを使っています。3C00H 番地がベクトルの最初の要素です。データ形式は、ADSP674 の実数フォーマットです。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストと ADSP674 のアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

このブロックは１つのブロック線図に１個しか配置できません。

２．３．４．　ホスト DP 出力デバイスの概念

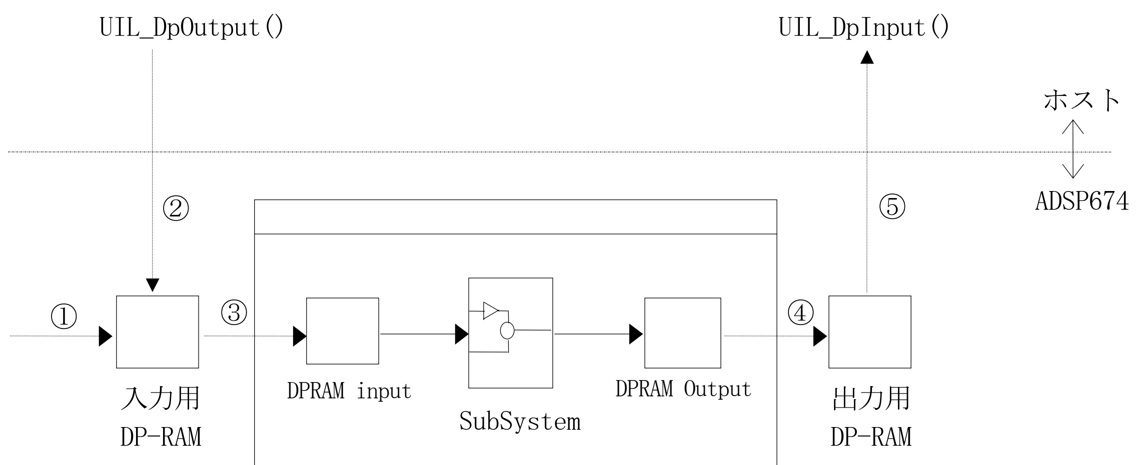
ホスト DPRAM インターフェースは、ADSP674 に実装されているデュアルポート RAM (DPRAM*[1参考]) を使い、ホストのユーザインターフェースライブラリと ADSP674 内の RTW によるリアルタイムモデルとの間で、非同期なデータ交換を可能にするものです。この機能を用いて相互にデータ転送が可能です。

この機能の実装は次の２つの作業で可能となります。

１、 ホスト側のユーザプログラムに UIL ライブラリをリンク。

２、 ADSP674 側の SIMUINK ブロック線図に DP ライブラリを配置。

次の図は DP-RAM インターフェースのデータ交換の流れを記した図です。



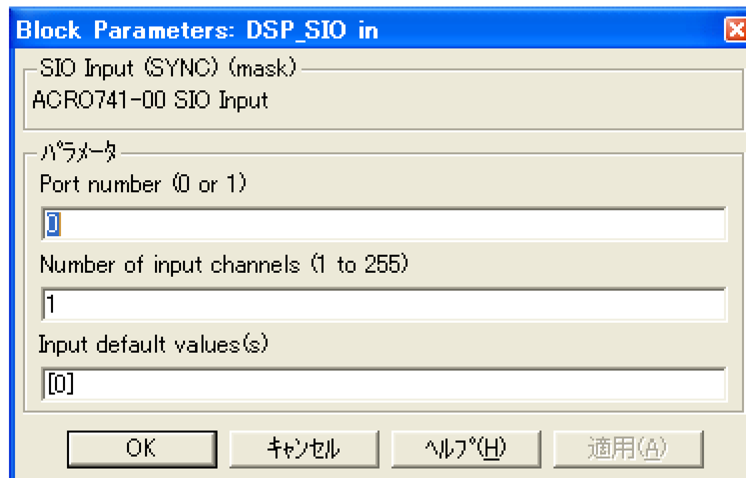
①リアルタイムモデルの初期化時にセットされます。
②ホストプログラムから関数呼び出し時にセットされます。
③ＲＴモデルのステップサイズ毎に読み出しされます。
④ＲＴモデルのステップサイズ毎に書き込まれます。
⑤ホストプログラムから関数呼び出し時点のデータが読み出されます。

＊１参考：デュアルポートＲＡＭ（ＤＰＲＡＭ）は ADSP674 ボードに標準実装されている特殊なメモリーで、ADSP674 とホストコンピュータの間で非同期、かつ、双方向に通信が可能です。

２．３．５．　SIO 入力デバイスドライバブロック

SIO 入力デバイスドライバブロックは、ADSP674 の SIO ポートを使い、他の ADSP674 と通信します。このブロックを使用するには、SIO 初期化ブロックを同じブロック線図に配置する必要があります。

図 ２．３−４　SIO in



● Number of input Channels (1 to 255)
受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

● Default Value(s)
相手が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数　(Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックから入力されるデータは、送信元 ADSP674 が送信した最も新しいデータです。複数チャンネルのデータを受け取る場合、チャンネル間のデータの同時性は保証されません。
受け取る側と相手側から送信されて来るデータのチャンネル数の整合は、確認されません。もし、相手 ADSP674 が送信したデータの点数が Number of input Channels の設定を超える場合、超えた分のデータは破棄されます。逆に、Number of input Channels で指定された点数のデータが送られない場合は、不足するチャンネルからは、Default Value(s) で設定したデフォルト値を出力します。例えば受信側が５チャンネルで送信側が３チャンネルの場合、先頭側の３チャンネルには相手から送られたデータが、そして残りは Default Value(s) で設定した値が入力されます。
データはベクトルで出力されますので、個々のチャンネルのスカラーデータが必要な場合は、Demux 等で分解して利用します。

２．３．６．　SIO 出力デバイスドライバブロック

SIO 出力デバイスドライバブロックは、ADSP674 の SIO ポートを使い、他の ADSP674（ADSP674）に対してデータを送信します。
このブロックを使用するには、SIO 初期化ブロックを同じブロック線図に配置する必要があります。

図　２．３－５　SIO out



- Number of output Channels (1 to 255)
  送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとをこのブロックに入力します。

このブロックに出力されるデータは、SIO ポートに自動的に送信されます。
このブロックへの出力データはベクトルで与えます。複数のスカラーデータを出力する場合は、Mux ブロック等でベクトルにまとめてから出力します。

２．３．７．　SIO 初期化デバイスドライバブロック

SIO 初期化デバイスドライバブロックは、SIO 入力・出力デバイスドライバブロックを使う場合に、SIO ポートを初期化する為使います。

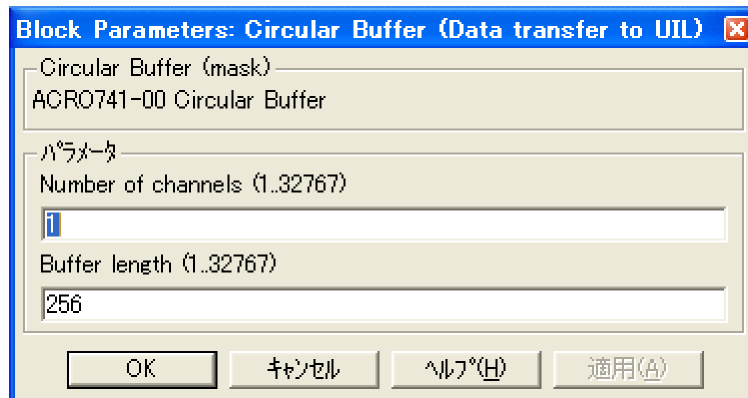
図　２．３−６　SIO Init



２．３．８．　サーキュラーバッファ・デバイスドライバブロック

ADSP674 のメモリ内に高速にデータ保管が行えるサーキュラーバッファを構築し、ここにデータを貯える為のブロックです。貯えたデータは UIL（オプション）にて取り出します。

図　２．３−７　C Buffer(74)



- Number of Channels (1 .. 32767)
  このブロックへの入力データの点数を指定します。入力データはベクトルで与えますので、このベクトルの幅と同じです。

- Buffer length (1 .. 32767)
  各データの保存する長さを指定します。

データバッファは ADSP674 内のメモリに確保されます。その概ねの長さは、

バッファ長さ　＝　（データ点数＋１）×（データの長さ＋α）

α はバッファの長さの最小単位が２のＮ乗である事から、その最小単位に切り上げる為の係数です。この領域は、実行時にヒープ領域から確保されますので、充分な長さのヒープ領域が必要です。バッファが確保出来たかどうかは、UIL で確認できます。バッファが確保出来なかった場合は、データの保管は行われません。その場合、データ保管する長さを減らす等の再試行は、ブロック線図の変更と RTW の再実行により行ってください。

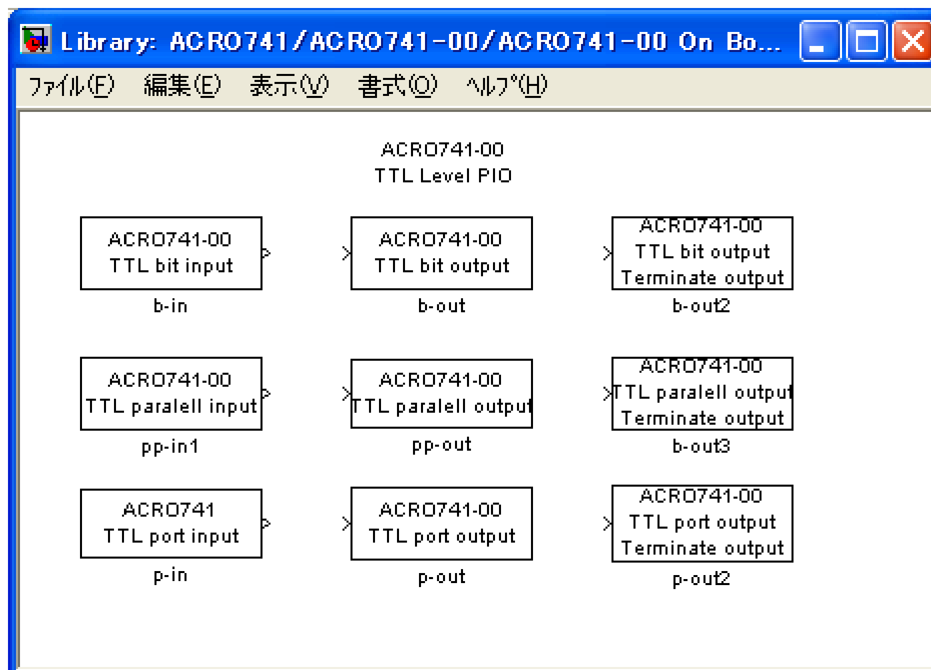
バッファは、サーキュラーバッファと呼ばれる、輪状につながったエンドレスのバッファを構築しています。バッファ内のデータが満杯になった段階で、新たなデータは最古のデータに上書きされ、最新の Buffer length 点のデータが、バッファに残ります。

データは UIL にて取り出します。現在何点のデータが格納されているか等の情報は、UIL で取り出すことができます。

２．４． ADSP674 On Board PIO ウィンドウ（ADSP674-00H/00 用）

ADSP674 ウィンドウの ADSP674 On Board PIO をダブルクリックして開くウィンドウです。ここには ADSP674 上に実装されている PIO ポートをアクセスする I/O デバイスドライバーが収納されています。

図 ２．４－１　ADSP674 On Board PIO



- ADSP674 TTL bit input
  １ビット単位で入力を行うブロックです。
- ADSP674 TTL bit output
  １ビット単位で出力を行うブロックです。
- ADSP674 TTL bit output Terminate output
  １ビット単位で出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。
- ADSP674 TTL parallel input
  全ビットをパラレルに入力するブロックです。
- ADSP674 TTL parallel output
  全ビットをパラレルに出力するブロックです。
- ADSP674 TTL parallel output Terminate output
  全ビットをパラレルに出力するブロックです。モデルの動作を終了する際に予め指定した状態を出力します。
- ADSP674 TTL port input
  ポート単位で入力を行うブロックです。
- ADSP674 TTL port output
  ポート単位で出力を行うブロックです。
- ADSP674 TTL port output Terminate output
  ポート単位で入力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

表　２．４－１　ビット番号と ADSP674PIO との対応

入出力信号名	ビット番号 注１	パラレル 注２	ポート単位 注３
EXTIN7/ EXTOUT7	7	最後	2^{7}
EXTIN6/ EXTOUT6	6	最後の１つ前	2^{6}
EXTIN5/ EXTOUT5	5	・	2^{5}
EXTIN4/ EXTOUT4	4	・	2^{4}
EXTIN3/ EXTOUT3	3	・	2^{3}
EXTIN2/ EXTOUT2	2	・	2^{2}
EXTIN1/ EXTOUT1	1	先頭の次	2^{1}
EXTIN0/ EXTOUT0	0	先頭	2^{0}

注１：ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

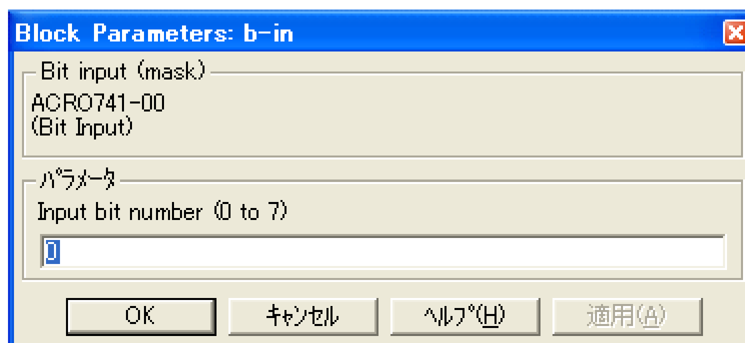
注２：パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注３：ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

２．４．１．　ADSP674 ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時０、HIGH の時１です。入力信号が接続されていないビットは１になります。

図 ２．４－２　ADSP674 TTL bit input

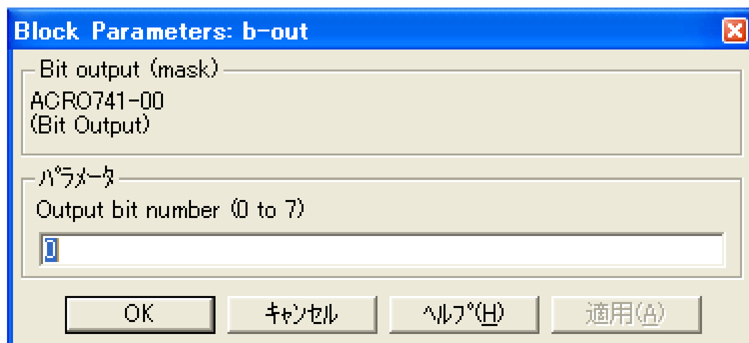


- Input bit number (0 to 7)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．４－１ (16ページ) を参照してください。

２．４．２.　ADSP674 ビット単位出力デバイスドライバブロック

　ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 ２．４－３　ADSP674 TTL bit output

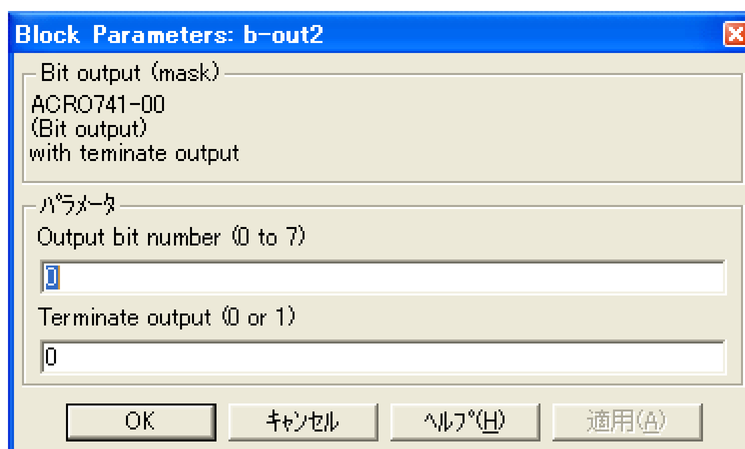


- Output bit number (0 to 7)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．４－１ (16ページ) を参照してください。

２．４．３．　ADSP674 終了時出力指定ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。
このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 ２．４－４　ADSP674 TTL bit output with Terminate output



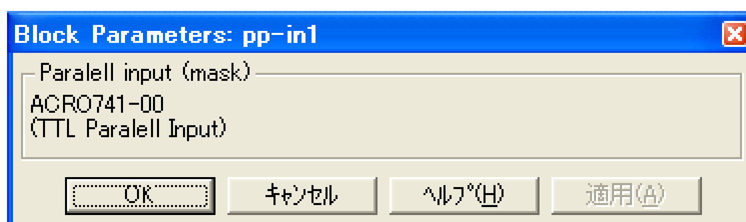
- Output bit number (0 to 7)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．４－１ (16ページ) を参照してください。

- Terminate output　(0 or 1)
  実時間モデルが停止する際に出力して欲しい状態を設定します。０で LOW, １で HIGH が出力されます。

２．４．４．　ADSP674 パラレル入力デバイスドライバブロック

パラレル入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの数と同じ点数のベクトルとして入力します。ボードに入力される信号レベルが LOW の時、これに該当する信号は０となります。入力が接続されていない信号の値は不定です。

図 ２．４－５　ADSP674 TTL paralell input



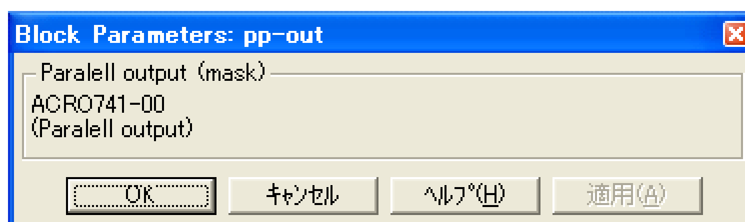
このブロックの初期化ダイアログには設定すべき項目はありません。
入力信号のビット位置とハードウェアとの対応は、表 ２．４－１（16ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

２．４．５．　ADSP674 パラレル出力デバイスドライバブロック

パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が 0 の場合は LOW を、それ以外の場合は HIGH を出力します。

図 ２．４－６　ADSP674 TTL paralell output



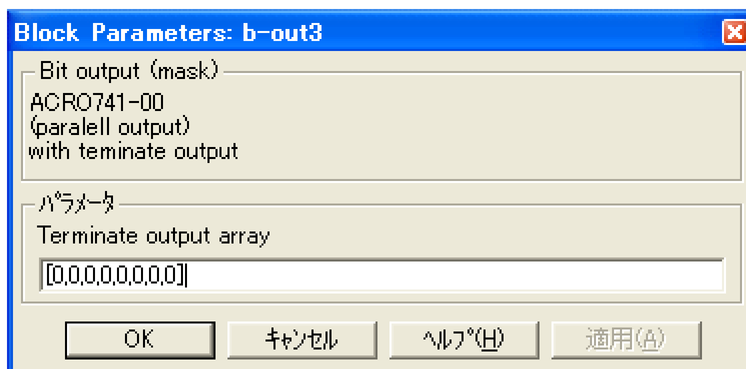
このブロックの初期化ダイアログには設定すべき項目はありません。
出力信号のビットの位置とハードウェアとの対応は、表 ２．４－１（16ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は､ビット単位出力デバイスドライバブロックにより失われます。

２．４．６．　ADSP674 終了時出力指定パラレル出力デバイスドライバブロック

終了時出力指定パラレル出力デバイスドライバブロックは、ブロック線図から出力ポートにのビット数と同じ点数のベクトルを受け取り、個々の要素に対応するビットに出力します。対応する要素の値が 0 の場合は LOW を、それ以外の場合は HIGH を出力します。
このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図　２．４－７　ADSP674 TTL paralell output



- Terminate output array
  実時間モデルが停止する際に出力して欲しい値をベクトルで指定します。

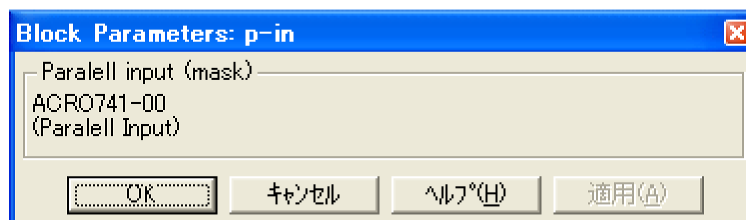
  出力信号のビットの位置とハードウェアとの対応は、表　２．４－１（16ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同時に使用できません。同時に使用した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

２．４．７．　ADSP674 ポート入力デバイスドライバブロック

ポート入力デバイスドライバブロックは、PIO ポートのデータを読み込み、ビットの重みをかけて加算された値を入力します。即ち PIO 入力をバイナリ値として読み込みます。ボードに入力される信号レベルが LOW の時、これに該当するビットは０となります。入力が接続されていない信号の値は不定です。

図　２．４−８　ADSP674 TTL port input

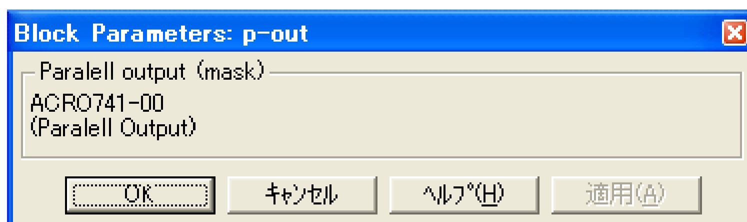


このブロックの初期化ダイアログには設定すべき項目はありません。
入力信号のビットの重みとハードウェアとの対応は、表　２．４−１（16ページ）を参照してください。

２．４．８．　ADSP674 ポート出力デバイスドライバブロック

　ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが０の場合、ボードに出力される信号レベルは LOW となります。
ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。

図　２．４－９　ADSP674 TTL port　output

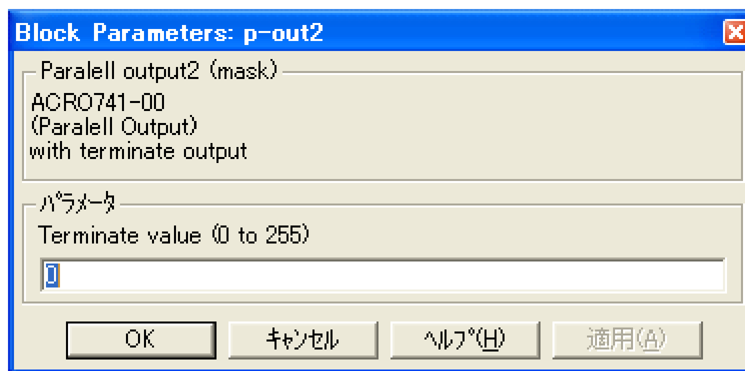


このブロックの初期化ダイアログには設定すべき項目はありません。
出力信号のビットの重みとハードウェアとの対応は、表　２．４－１（16ページ）を参照してください。

２．４．９．　ADSP674 終了時出力指定 ポート出力デバイスドライバブロック

終了時出力指定ポート出力デバイスドライバブロックは、ブロック線図から受け取った値をバイナリ値に変換し、PIO ポートのへ出力します。バイナリに変換後のビットが０の場合、ボードに出力される信号レベルは LOW となります。
ラップラウンドによるクリッピングは行っていません。ブロック線図上で値の上下限を規制してください。
このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図　２．４－１０　ADSP674 TTL port　output　terminate output



- Terminate value (0 to 255)
  実時間モデルが停止する際に出力して欲しい値を１０進数で設定します。

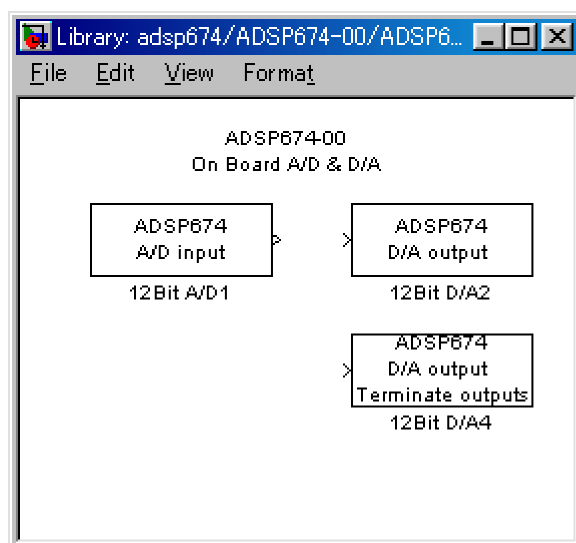
  出力信号のビットの重みとハードウェアとの対応は、表　２．４－１（16ページ）を参照して下さい。

２．５．　ADSP674-00 On Board Ａ／Ｄ＆Ｄ／Ａ ウィンドウ（ADSP674-00H/00 用）

２．５．１．　概要

ADSP674-00 ウィンドウの　ADSP674-00 On Board A/D & D/A をダブルクリックして開くウィンドウです。ここにはDSP ボード上に実装されている AD や DA ポートをアクセスする IO デバイスドライバーが収納されています。

図　２．５−１　ADSP674-00 On Board AD & DA ドライバウィンドウ



- ADSP674-00 AD input

  AD から入力を行うブロックです。
- ADSP674-00 DA output

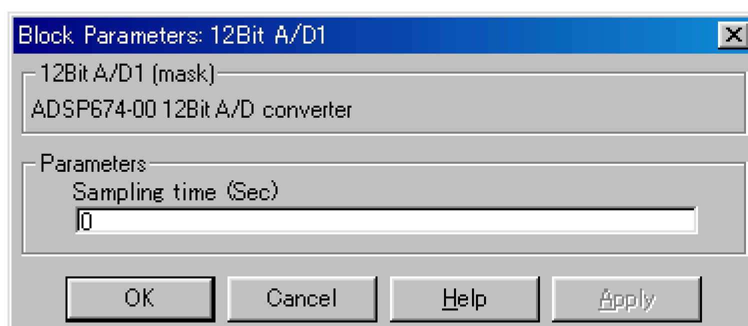
  DA への出力を行うブロックです。
- ADSP674-00 DA output Terminate output

  DA への出力を行うブロックです。モデルの動作を終了する際に予め指定した状態を出力します。

２．５．２．　ADSP674-00 Ａ／Ｄ入力デバイスドライバーブロック

Ａ／Ｄ入力デバイスドライバーブロックは、DSP ボード上のＡ／Ｄから一括してデータを入力します。入力の対象となるＡ／Ｄチャンネルは､このブロックからの出力は､２の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。
このブロックからの出力データの単位は電圧値［Ｖ］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

図 ２．５-2　ADSP674-00 A/D input



- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は､ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

２．５．３．　ADSP674-00　Ｄ／Ａ出力デバイスドライバーブロック

Ｄ／Ａ出力デバイスドライバーブロックは、幅２のベクトルデータを受取り、Ｄ／Ａへ一括して出力します。このブロックへの入力ベクトルの幅は必ず２でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のＤ／Ａへ出力されます。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．５-3　ADSP674-00 D/A output

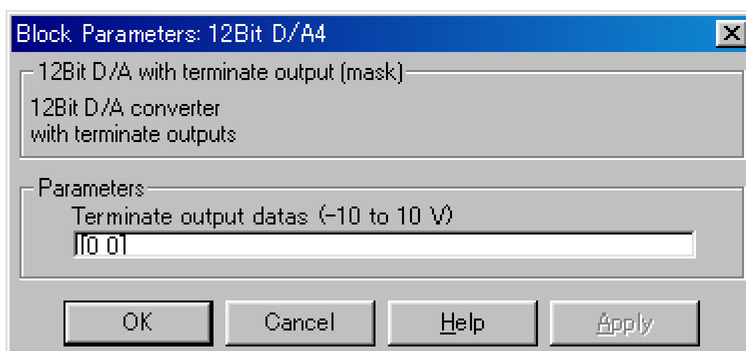


このブロックの初期化ダイアログには設定すべき項目はありません。

２．５．４．　ADSP674-00 終了時出力指定Ｄ／Ａ出力デバイスドライバーブロック

終了時出力指定Ｄ／Ａ出力デバイスドライバーブロックは、幅２のベクトルデータを受取り、Ｄ／Ａへ一括して出力します。このブロックへの入力ベクトルの幅は必ず２でなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０のＤ／Ａへ出力されます。
このブロックはモデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[Ｖ]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．５-4　ADSP674-00 D/A output



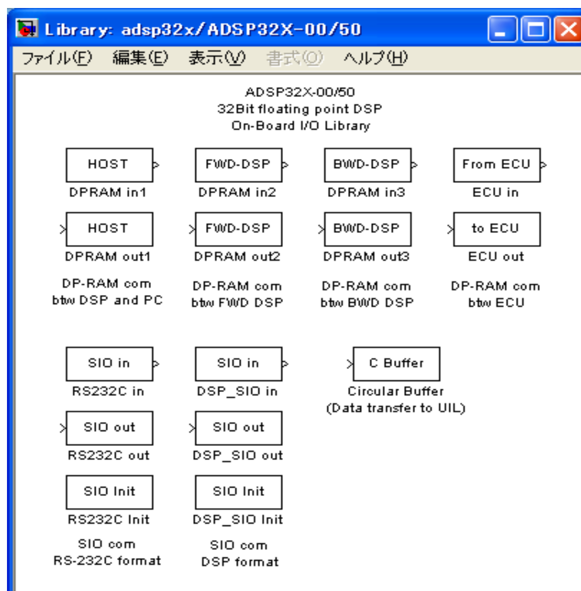
- Terminate output datas　(-10 to 10V)
  リアルタイムモデルが終了する際にＤ／Ａから出力する値をベクトルで設定します。単位は[Ｖolt]です。

２．６．　ADSP32X-00/50 デバイスドライバーブロック（ADSP324-00A 用）

２．６．１．　概要

図　２．３−１に、ADSP324-00A（ＤＳＰボード）用デバイスドライバウィンドウを開いた所を示します。

図　２．６−１　ADSP32X-00/50 ウィンドウ



この中には ＤＳＰボード上のＩ／Ｏ（ＤＰＲＡＭ等含む）とのインターフェースを取るためのデバイスドライバーブロックが登録されています。パイプライン＆ＲＡＭボード（ADSP32X-07/57 又は、-08/58 系ボード）に関するブロックもこの中に含まれています。

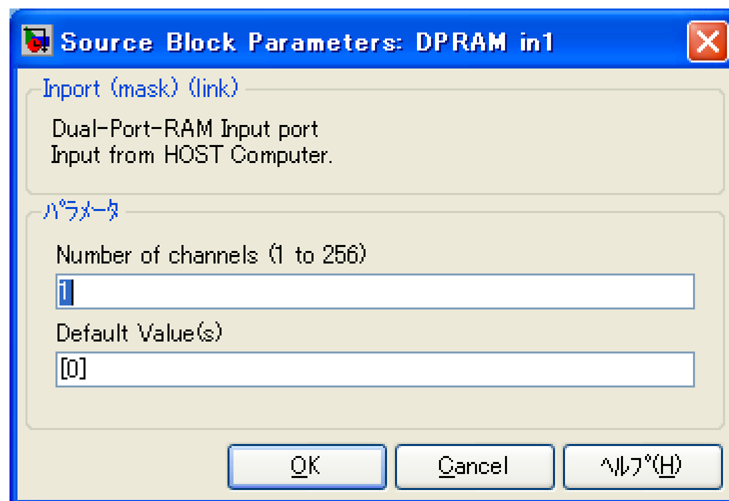
HOST	ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストとの間で非同期にデータ交換をする為のブロックです。
FWD-DSP	前段（注１）ＤＳＰとの間にあるＤＰＲＡＭを使い、前段ＤＳＰとの間で非同期にデータ交換をする為のブロックです。
BWD-DSP	後段（注１）ＤＳＰとの間にあるＤＰＲＡＭを使い、後段ＤＳＰとの間で非同期にデー交換をする為のブロックです。
From ECU	前段（注１）ＤＳＰとの間にあるＤＰＲＡＭを使い、ＥＣＵ等の外部ＣＰＵとの間でデータ交換を行う為のブロックです。
To ECU	前段（注１）ＤＳＰとの間にあるＤＰＲＡＭを使い、ＥＣＵ等の外部ＣＰＵとの間でデータ交換を行う為のブロックです。
SIO in ／ SIO out	ＤＳＰのＳＩＯポートを使い、シリアル通信を行う為のブロックです。ＲＳ２３２Ｃフォーマットによる外部ＣＰＵ通信と、ＤＳＰ固有のフォーマットによるＤＳＰ間通信が、いずれか一方のみ選択的に利用出来ます。
SIO Init	ＤＳＰのＳＩＯポートを使い、シリアル通信を行う際に、ＳＩＯポートの初期化を行う為のブロックです。
C Buffer	ＤＳＰのサーキュラーバッファ機構を使った高速データバッファリングによるホストへのデータ送信を行うブロックです。高速に処理させたいブロック線図で使用する事により、データをバッファリングする為の負担を低減させることが出来ます。バッファリングしたデータは、ＵＩＬを用いたユーザアプリケーションへデータで取り出す事が出来ます。

注１：前段・後段ＤＳＰについては、“ADSP674 の前段と後段の意味”（123ｐ）を参照してください。

２．６．２．　ホストＤＰ入力デバイスドライバーブロック

ホストＤＰ入力デバイスドライバーブロックは、ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストーＤＳＰ間で非同期にデータ交換を行う為のブロックです。主に、ＵＩＬを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、0
ホストＤＰ入出力デバイスの概念 を参照してください。

図　２．６-2 HOST DPRAM in



- Number of Channels (1 to 256)
  ホストから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)
  ホストが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数　(Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

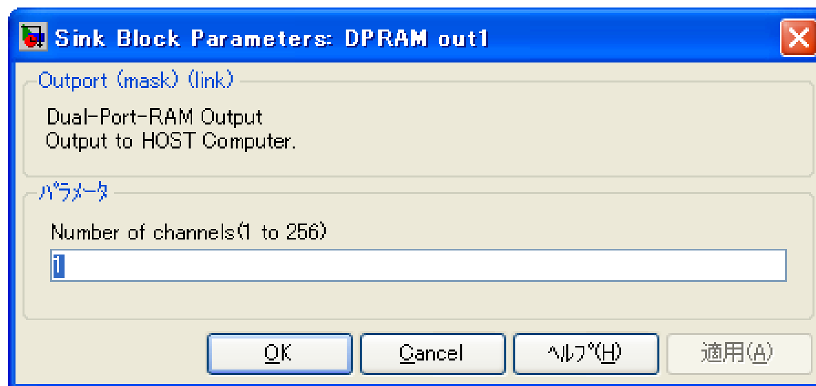
このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２００６００Ｈ番地から連続する２５６ワードを使っています。２００６００Ｈ番地がベクトルの最初の要素です。データ形式は、ＤＳＰの実数フォーマットですので、ＵＩＬ以外のアプリケーションで通信する場合はホスト側でＩＥＥＥとの間のデータ変換が必要です。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとＤＳＰのアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

このブロックは１つのブロック線図に１個しか配置できません。

２．６．３．　ホストＤＰ出力デバイスドライバーブロック

ホストＤＰ出力デバイスドライバーブロックは、ＤＳＰとホストとの間にあるＤＰＲＡＭを使い、ホストーＤＳＰ間で非同期にデータ交換を行う為のブロックです。主に、ＵＩＬを使って作成したホスト側アプリケーションで、比較的ゆっくりと変化をするデータや、時々しか変更しないパラメータ等を、ブロック線図上のデバイスと直接交換する場合等に使用します。通信の概要は、0
ホストＤＰ入出力デバイスの概念 を参照してください。

図　２．６-3 HOST DPRAM out



- Number of Channels (1 to 256)
  ホストへ送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルにまとめてからこのブロックに入力します。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２００７００Ｈ番地から連続する２５６ワードを使っています。２００７００Ｈ番地がベクトルの最初の要素です。データ形式は、ＤＳＰの実数フォーマットですので、ＵＩＬ以外のアプリケーションで通信する場合はホスト側でＩＥＥＥとの間のデータ変換が必要です。
複数のデータを渡す場合、データの同時性は保証されません。つまり、ホストとＤＳＰのアクセスがほぼ同時に起こった場合、データの前半と後半で１時刻分ずれる可能性があります。

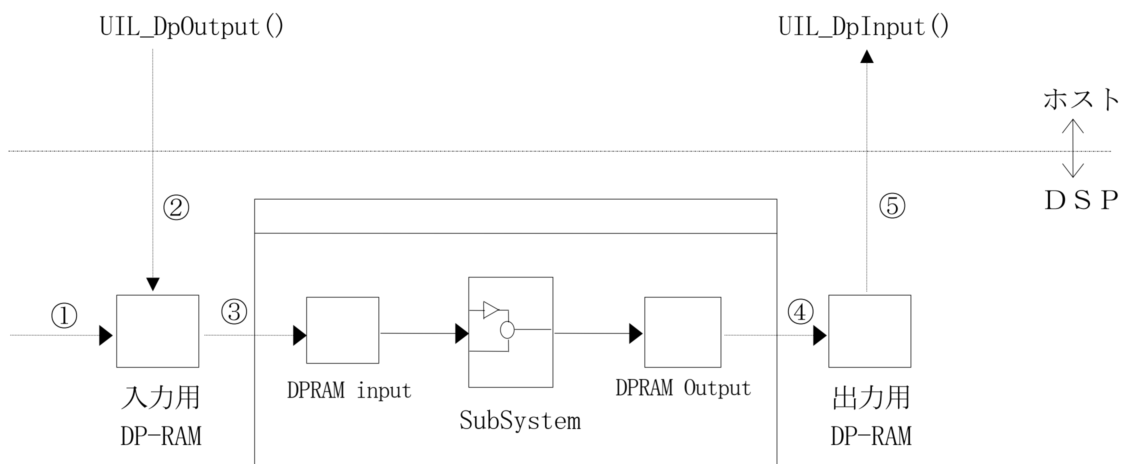
このブロックは１つのブロック線図に１個しか配置できません。

２．６．４．　ホストＤＰ入出力デバイスの概念

ホストＤＰＲＡＭインターフェースは、ＤＳＰボードに実装されているデュアルポートＲＡＭ（ＤＰＲＡＭ*[1参考]）を使い、ホストのユーザインターフェースライブラリとＤＳＰ内のＲＴＷによるリアルタイムモデルとの間で、非同期なデータ交換を可能にするものです。この機能を用いて相互にデータ転送が可能です。

この機能の実装は次の２つの作業で可能となります。
３、ホスト側のユーザプログラムにＵＩＬライブラリをリンク。
４、ＤＳＰ側のＳｉｍｕｌｉｎｋブロック線図にＤＰライブラリを配置。

次の図はＤＰ－ＲＡＭインターフェースのデータ交換の流れを記した図です。



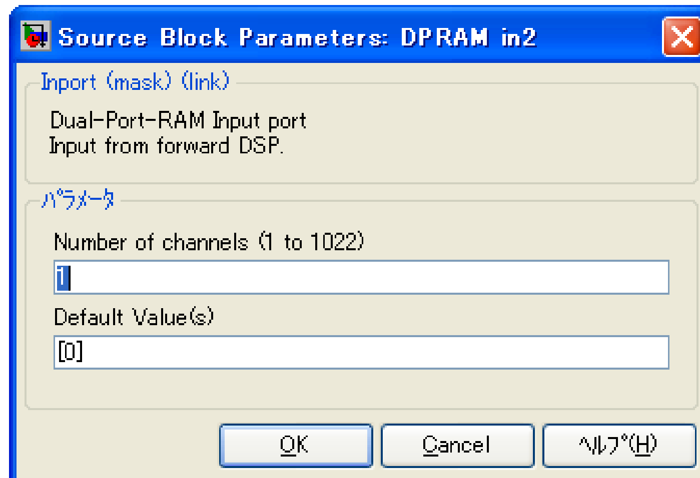
①リアルタイムモデルの初期化時にセットされます。
②ホストプログラムから関数呼び出し時にセットされます。
③ＲＴモデルのステップサイズ毎に読み出しされます。
④ＲＴモデルのステップサイズ毎に書き込まれます。
⑤ホストプログラムから関数呼び出し時点のデータが読み出されます。

＊１参考：デュアルポートＲＡＭ（ＤＰＲＡＭ）はＤＳＰボードに標準実装されている特殊なメモリーで、ＤＳＰとホストコンピュータの間で非同期、かつ、双方向に通信が可能です。

２．６．５．　前段ＤＰ入力デバイスドライバーブロック

前段ＤＰ入力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いてＤＳＰ間で非同期にデータ交換を行うためのブロックです。複数のＤＳＰボードを用いてマルチＤＳＰシステムを構築する場合に有用です。

図　２．６-4　FWD DSP DPRAM in



- Number of Channels (1 to 1022)
  相手ＤＳＰから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)
  相手ＤＳＰが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数　(Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２００８００Ｈ番地から連続する１Ｋワードを使っています。先頭の２ワードはアービトレーションに使用しており、残りの１０２２ワードがデータです。

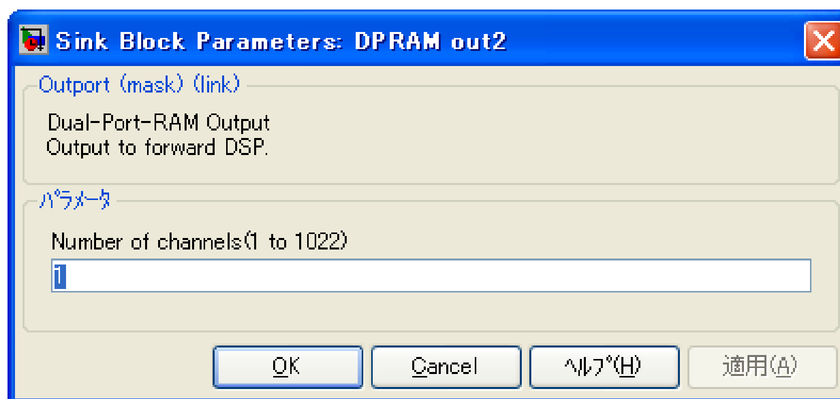
複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、
“ADSP674 間DP通信アービトレーション”（122p）を参照してください。
又、前段・後段ＤＳＰについては、“ADSP674 の前段と後段の意味”（123ｐ）を参照してください。

このブロックは１つのブロック線図に１個しか配置できません。

２．６．６．　前段ＤＰ出力デバイスドライバーブロック

前段ＤＰ出力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いてＤＳＰ間で非同期にデータ交換を行うためのブロックです。複数のＤＳＰボードを用いてマルチＤＳＰシステムを構築する場合に有用です。

図　２．６-5 FWD DSP DPRAM out



- Number of Channels (1 to 1022)
  相手ＤＳＰに渡すデータのチャンネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２００Ｃ００Ｈ番地から連続する１Ｋワードを使っています。先頭の２ワードはアービトレーションに使用しており、残りの１０２２ワードがデータです。

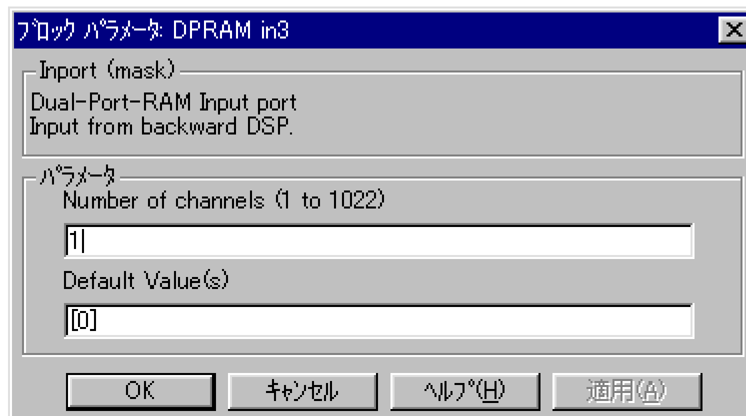
複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、
“ADSP674 間DP通信アービトレーション”（122p）を参照してください。
又、前段・後段ＤＳＰについては、“ADSP674 の前段と後段の意味”（123ｐ）を参照してください。

このブロックは１つのブロック線図に１個しか配置できません。

２．６．７．　後段ＤＰ入力デバイスドライバーブロック

後段ＤＰ入力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いてＤＳＰ間で非同期にデータ交換を行うためのブロックです。複数のＤＳＰボードを用いてマルチＤＳＰシステムを構築する場合に有用です。

図　２．６-6　BWD DSP DPRAM in



- Number of Channels (1 to 1022)

  相手ＤＳＰから受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)

  相手ＤＳＰが未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２０１０００Ｈ番地から連続する１Ｋワードを使っています。先頭の２ワードはアービトレーションに使用しており、残りの１０２２ワードがデータです。

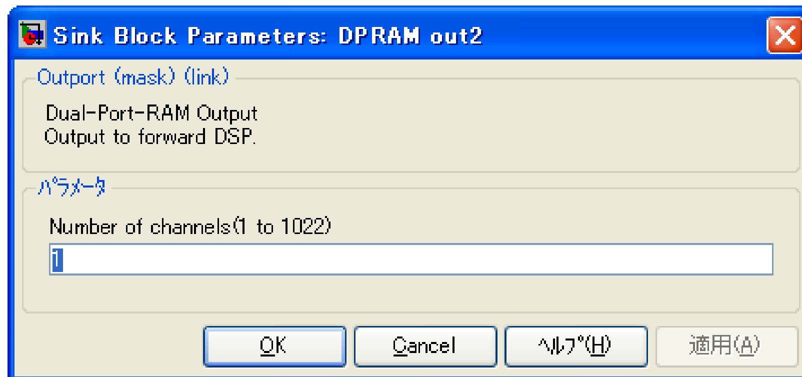
複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、
“ADSP674 間DP通信アービトレーション”（122p）を参照してください。
又、前段・後段ＤＳＰについては、“ADSP674 の前段と後段の意味”（123ｐ）を参照してください。

このブロックは１つのブロック線図に１個しか配置できません。

２．６．８．　後段ＤＰ出力デバイスドライバーブロック

後段ＤＰ出力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いてＤＳＰ間で非同期にデータ交換を行うためのブロックです。複数のＤＳＰボードを用いてマルチＤＳＰシステムを構築する場合に有用です。

図　２．６-7　BWD DSP DPRAM out



- Number of Channels (1 to 1022)
  相手ＤＳＰに渡すデータのチャンネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

このブロックが通信に使用する領域は、ＤＳＰから見たメモリーマップの、２０１４００Ｈ番地から連続する１Ｋワードを使っています。先頭の２ワードはアービトレーションに使用しており、残りの１０２２ワードがデータです。

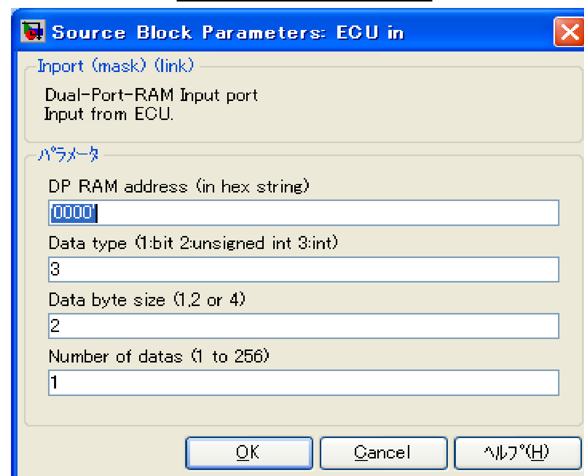
複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、
“ADSP674 間DP通信アービトレーション”（122p）を参照してください。
又、前段・後段ＤＳＰについては、“ADSP674 の前段と後段の意味”（123ｐ）を参照してください。

このブロックは１つのブロック線図に１個しか配置できません。

２．６．９．　ＥＣＵ入力デバイスドライバーブロック

ＥＣＵ（External Computer Unit）入力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いて、ＤＳＰと外部ＣＰＵとの間でデータ交換を行うためのブロックです。ＤＳＰボードを用いて実現した仮想実機や、プロトタイピングを外部ＣＰＵと接続する場合に有用です。このブロックを使用する場合は、パイプラインボードの他に、パイプラインボードと外部ＣＰＵとを接続する為の“バスＩ／Ｆハードウェア”が別途必要です。

図　２．６-8 From ECU



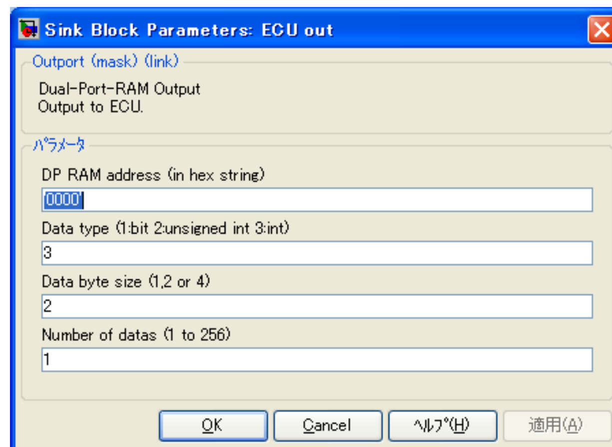
- DP RAM address (in hex string)
  ＤＰＲＡＭの通信に用いる領域先頭からの番地オフセットを設定します。１６進数の文字列で指定してください。通常は偶数番地を設定します。外部ＣＰＵから見たメモリーマップは、ＤＰＲＡＭと外部ＣＰＵのバスと間に配置する“バスＩ／Ｆハードウェア”に依存します。詳しくは、Ｉ／Ｆハードウェアの説明書を参照してください。
- Data type (1:bit 2:unsigned int 3:int)
  データの型を指定します。１:bit は１外部ＣＰＵから見た１つのビットを、ブロック線図上で１つのスカラーデータとして取り扱います。２:unsigned int はデータを無符号の整数として取り扱います。３:int はデータを符号付き整数として取り扱います。
- Data byte size (1,2 or 4)
  データのバイトサイズを指定します。
- Number of datas (1 to 256)
  このブロックが取り扱うデータの点数を指定します。

リアルタイムモデルの場合、ＤＳＰ内でのデータの取り扱いは全て実数で行われます。ＤＳＰ内の実数と外部ＣＰＵの整数との変換は、デバイスドライバ内で行われます。
外部ＣＰＵのデータ配置とＤＳＰとのデータの関係は　“0
外部ＣＰＵ（ECU）とのデータ対応　(52p)”を参照してください。
このブロックは１つのブロック線図の中に、必要に応じて複数のブロックを配置する事ができますが、DPRAM address の設定が妥当でないと、アドレスが重複してしまいます。これに対するチェックは行われていません。充分注意して使用してください。

２．６．１０．ＥＣＵ出力デバイスドライバーブロック

ＥＣＵ (External Computer Unit) 出力デバイスドライバーブロックは、ＤＳＰとＤＳＰを接続するパイプラインボード（オプション）を用いて、ＤＳＰと外部ＣＰＵとの間でデータ交換を行うためのブロックです。ＤＳＰボードを用いて実現した仮想実機や、プロトタイピングを外部ＣＰＵと接続する場合に有用です。このブロックを使用する場合は、パイプラインボードの他に、パイプラインボードと外部ＣＰＵとを接続する為の“バスＩ／Ｆハードウェア”が別途必要です。

図　２．６-9 To ECU



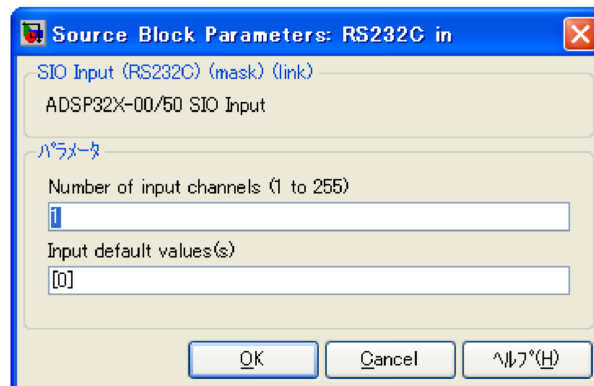
- DP RAM address (in hex string)
  ＤＰＲＡＭの通信に用いる領域先頭からの番地オフセットを設定します。１６進数の文字列で指定してください。通常は偶数番地を設定します。外部ＣＰＵから見たメモリーマップは、ＤＰＲＡＭと外部ＣＰＵのバスと間に配置する“バスＩ／Ｆハードウェア”に依存します。詳しくは、Ｉ／Ｆハードウェアの説明書を参照してください。
- Data type (1:bit 2:unsigned int 3:int)
  データの型を指定します。１:bit は１外部ＣＰＵから見た１つのビットを、ブロック線図上で１つのスカラーデータとして取り扱います。２:unsigned int はデータを無符号の整数として取り扱います。３:int はデータを符号付き整数として取り扱います。
- Data byte size (1,2 or 4)
  データのバイトサイズを指定します。
- Number of datas (1 to 256)
  このブロックが取り扱うデータの点数を指定します。

リアルタイムモデルの場合、ＤＳＰ内でのデータの取り扱いは全て実数で行われます。ＤＳＰ内の実数と外部ＣＰＵの整数との変換は、デバイスドライバ内で行われます。
外部ＣＰＵのデータ配置とＤＳＰとのデータの関係は　“0
外部ＣＰＵ (ECU) とのデータ対応　(52p)”を参照してください。
このブロックは１つのブロック線図の中に、必要に応じて複数のブロックを配置する事ができますが、DPRAM address の設定が妥当でないと、アドレスが重複してしまいます。これに対するチェックは行われていません。充分注意して使用してください。

２．６．１１．ＳＩＯ(RS232-C)入力デバイスドライバーブロック

ＳＩＯ(RS232-C)入力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使い外部ＣＰＵとの間で RS232-C フォーマットによる通信を用いたデータ交換を行います。このブロックを用いて RS232-C 機器と通信するには、Ｒ２３２－Ｃ変換ユニットＡＤＳＰ３２４－１２５（オプション）が必要です。このブロックは、ＳＩＯ(DSP)入力／ＳＩＯ(DSP)出力等と同時には使えません。
このブロックを使用するには、ＳＩＯ(RS232-C)初期化ブロックを同じブロック線図に配置する必要があります。

図　２．６-10　RS232-C in



- Number of input Channels (1 to 255)
  受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)
  相手が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数 (Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックから入力されるデータは、送信元ＲＳ２３２－Ｃ機器が送信した最も新しいデータです。複数チャンネルのデータを受け取る場合、チャンネル間のデータの同時性は保証されません。

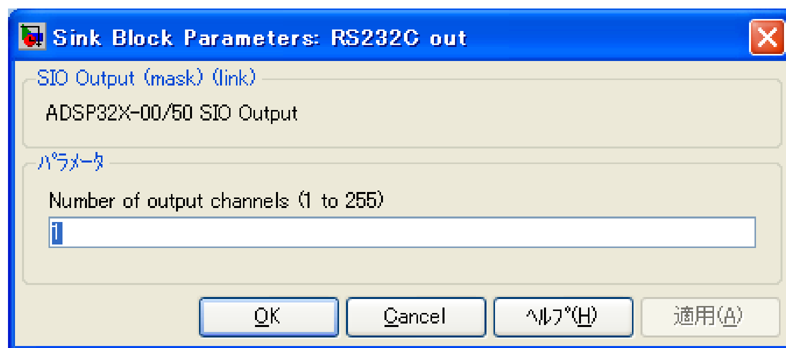
データチャンネルと、外部機器との通信フォーマットについては、“0
ＳＩＯ(RS232-C)通信フォーマット”(43p) を参照してください。
外部機器からのデータのチャンネル番号が Number of input Channels の設定を超える場合、そのデータは破棄されます。逆に、Number of input Channels で指定されたチャンネル番号のデータが送られない場合は、Default Value(s) で設定したデフォルト値を出力します。

データはベクトルで出力されますので、個々のチャンネルのスカラーデータが必要な場合は、Demux 等で分解して利用します。データチャンネル番号１番がベクトルの最初の要素に相当します。

２．６．１２．ＳＩＯ(RS232-C)出力デバイスドライバーブロック

ＳＩＯ(RS232-C)入力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使い外部ＣＰＵとの間で RS232-C フォーマットによる通信を用いたデータ交換を行います。このブロックを用いて RS232-C 機器と通信するには、Ｒ２３２－Ｃ変換ユニット ADSP324-125（オプション）が必要です。
このブロックは、ＳＩＯ(DSP)入力／ＳＩＯ(DSP)出力等と同時には使えません。
このブロックを使用するには、ＳＩＯ(RS232-C)初期化ブロックを同じブロック線図に配置する必要があります。

図　２．６-11　RS232C out



- Number of output Channels (1 to 255)
  送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとをこのブロックに入力します。

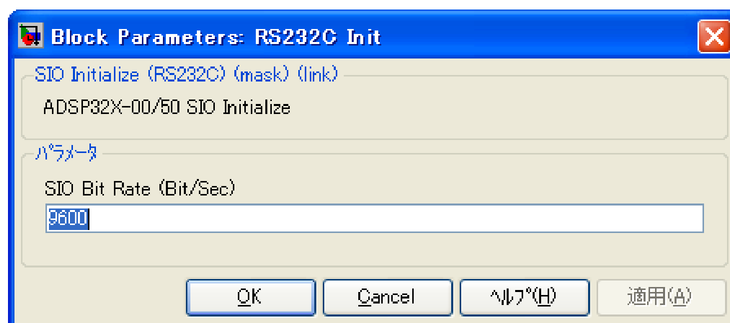
このブロックに出力されるデータは、ＤＳＰ内のメモリーに一旦保存され、外部機器が送信した“入力リクエスト”により読み出されます。外部機器が読み出す前にリアルタイムモデルが次のステップの計算を実行した場合、前回のデータは読み出されないまま上書きにより消失します。
ＤＳＰのリアルタイムモデル計算と、外部機器とは非同期です。よって、複数チャンネルのデータを送る場合、チャンネル間のデータの同時性は保証されません。

データチャンネルと、外部機器との通信フォーマットについては、“0
ＳＩＯ(RS232-C)通信フォーマット”（43p）を参照してください。
外部機器からのデータのチャンネル番号が Number of output Channels の設定を超える場合、そのデータは不定です。
このブロックへの出力データはベクトルで与えます。複数のスカラーデータを出力する場合は、Mux ブロック等でベクトルにまとめてから出力します。この時、データチャンネル番号１番がベクトルの最初の要素に相当します。

２．６．１３．ＳＩＯ(RS232-C)初期化デバイスドライバーブロック

ＳＩＯ(RS232-C)初期化デバイスドライバーブロックは、ＳＩＯ(RS232-C)入力･出力デバイスドライバーブロックを使う場合に、ＳＩＯを初期化する為使います。
このブロックは、ＳＩＯ(DSP)入力／ＳＩＯ(DSP)出力等と同時には使えません。

図　２．６-12　RS232-C Init



- SIO Bit Rate (Bit / Sec)
  ＳＩＯポートの転送速度（ボーレート）を設定します。外部機器の転送速度と同じでないといけません。

２．６．１４．ＳＩＯ(RS232-C)通信フォーマット

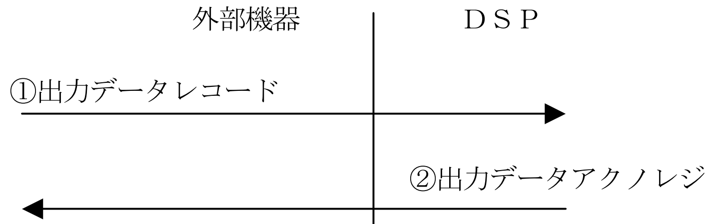
　ＤＳＰにはＳＩＯポートが１ポートしか有りません。ここを使い複数のデータを受け渡す為、データを識別する論理チャンネル番号を想定します。チャンネル数は最大２５５チャンネルです。各チャンネルには、１から２５５の番号が割り振られます。外部機器との間は、この識別番号とデータとをセットにして送信します。
ＤＳＰと外部機器との間は、以下の仕様で通信します。

＜通信仕様＞

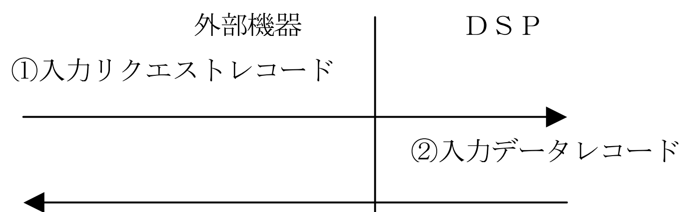
通信パラメータ	８ビット、パリティ無し、２ストップビット
通信フロー制御	無し
通信手順	半二重通信によるマスタースレーブ方式 （無手順）外部機器がマスター
通信速度	９６００BPS（ＭＡＸ）任意
入力（ホスト→ＤＳＰ）データ数	１～２５５チャンネル
出力（ＤＳＰ→ホスト）データ数	１～２５５チャンネル
入出力データ精度	３２ビット単精度実数
データフォーマット	識別コードとＩＥＥＥ７５４ﾌｫｰﾏｯﾄ(ﾊﾞｲﾅﾘｰ)

＜通信手順＞

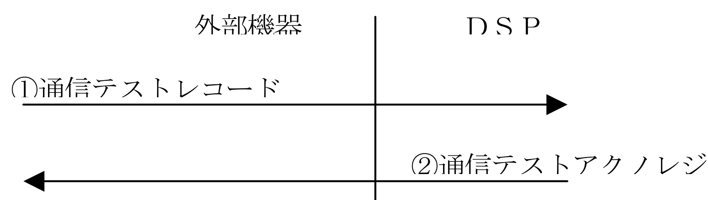
● 入力ブロック（外部機器→ＤＳＰ）



● 出力ブロック（外部機器←ＤＳＰ）



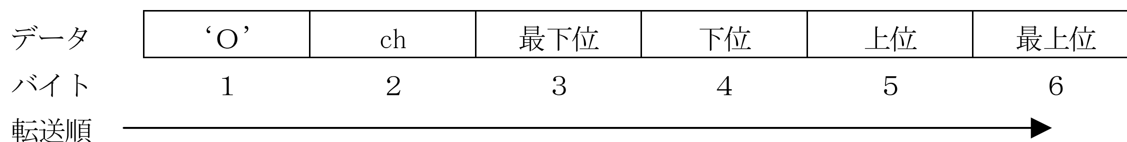
● 通信テスト（外部機器←ＤＳＰ）



＜レコードフォーマット＞

● 出力データレコード

外部機器からＤＳＰに対してデータを出力します。

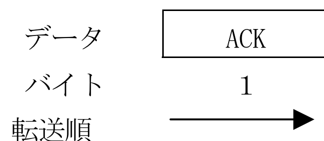


詳細

バイト	
1	レコード識別記号。半角 ASCII の大文字の‘O’（オー）。
2	チャンネル番号を表すバイナリ値。１～２５５
3	データ IEEE-754 単精度実数形式の最下位バイト
4	データ IEEE-754 単精度実数形式の下位バイト
5	データ IEEE-754 単精度実数形式の上位バイト
6	データ IEEE-754 単精度実数形式の最上位バイト

● 出力データアクノレジ

ＤＳＰが外部機器の出力データレコードに対して返す応答です。

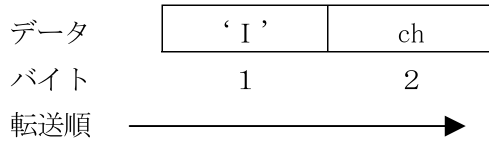


詳細

バイト	
1	レコード識別記号。ACK コード（バイナリー06H)。

- 入力リクエストレコード

外部機器がＤＳＰに対してデータの送出を要求します。

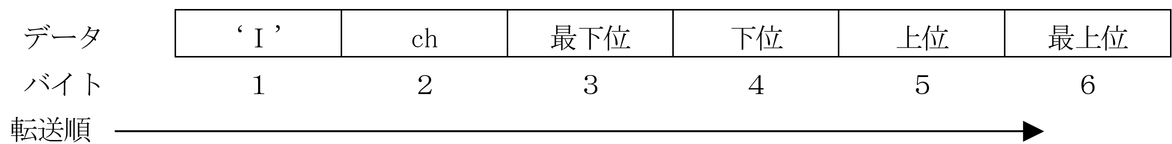


詳細

バイト	
1	レコード識別記号。半角 ASCII の大文字の‘Ｉ’（アイ）。
2	チャンネル番号を表すバイナリ値。１～２５５

- 入力データレコード

ＤＳＰが外部機器から送られた入力リクエストに対して返すデータレコードです。

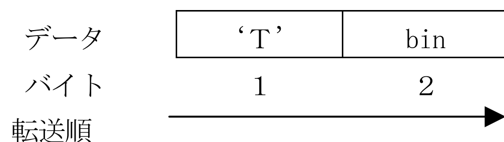


詳細

バイト	
1	レコード識別記号。半角 ASCII の大文字の‘Ｉ’（アイ）。
2	チャンネル番号を表すバイナリ値。１～２５５
3	データ IEEE-754 単精度実数形式の最下位バイト
4	データ IEEE-754 単精度実数形式の下位バイト
5	データ IEEE-754 単精度実数形式の上位バイト
6	データ IEEE-754 単精度実数形式の最上位バイト

- 通信テストレコード

外部機器がＤＳＰに対して、通信のテストの目的で送ります。

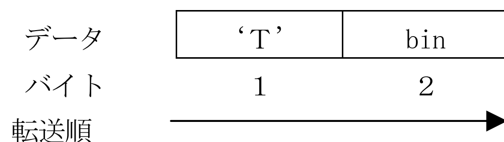


詳細

バイト	
1	レコード識別記号。半角 ASCII の大文字の‘T’(ティー)。
2	テストの為の任意のバイナリ値。０～２５５

- 通信テストアクノレジ

ＤＳＰが外部機器から送られた通信テストレコードに対して返す応答です。



詳細

バイト	
1	レコード識別記号。半角 ASCII の大文字の‘T’(ティー)。
2	通信テストレコードで受け取ったバイナリ値。

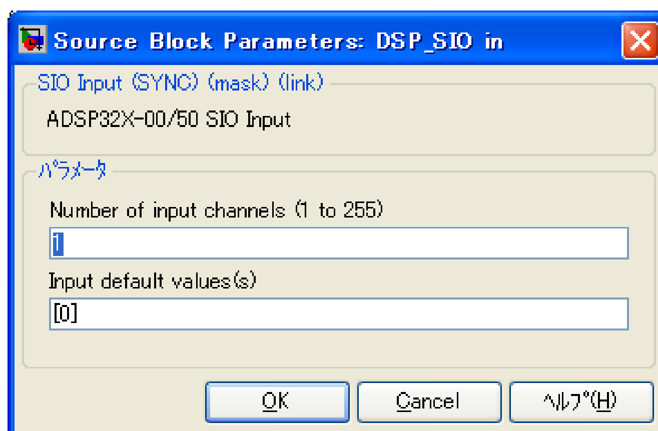
- 不良通信

　ホストからＤＳＰへ送信されるデータの第一バイトが、Ｏ、Ｉ、Ｔ、以外の場合は無視されます。又、正常な第一バイトに始まるレコードでも、１レコード完了までの間に通信オーバーラン等のエラーが発生した場合、当該レコードは破棄されます。

２．６．１５．ＳＩＯ(DSP)入力デバイスドライバーブロック

ＳＩＯ(DSP)入力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使って、ＤＳＰとＤＳＰの間で非同期にデータ交換を行う為のブロックです。このブロックへの出力データは、相手ＤＳＰのブロック線図に配置したＳＩＯ(DSP)出力デバイスドライバーブロックへ出力されたデータを読み取ることができます。
このブロックを使用するには、同じブロック線図には、ＳＩＯ(DSP)初期化デバイスドライバーブロックが必要です。又、このブロックは、ＳＩＯ(RS232C)入力（出力）ブロックと同時に使用する事はできません。

図 ２．６-13　DSP_SIO in



- Number of input Channels (1 to 255)
  受け取るデータのチャンネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。
- Default Value(s)
  相手が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第一引数　(Number of Channels) で指定した点数の幅を持つ定数ベクトルを与えます。

このブロックから入力されるデータは、相手ＤＳＰが送信した最も新しいデータです。複数チャンネルのデータを受け取る場合、チャンネル間のデータの同時性は保証されません。

チャンネル番号の対応等については、“0
ＳＩＯ(DSP)出力デバイスドライバーブロック”(48p) を参照してください。

２．６．１６．ＳＩＯ(DSP)出力デバイスドライバーブロック

ＳＩＯ(DSP) 出力デバイスドライバーブロックは、ＤＳＰのＳＩＯポートを使って、ＤＳＰとＤＳＰの間で非同期にデータ交換を行う為のブロックです。このブロックへの出力データは、相手ＤＳＰのブロック線図に配置したＳＩＯ(DSP)入力デバイスドライバーブロックで読み取ることができます。
このブロックを使用するには、同じブロック線図には、ＳＩＯ(DSP)初期化デバイスドライバーブロックが必要です。又、このブロックは、ＳＩＯ(RS232C)入力（出力）ブロックと同時に使用する事はできません。

図　２．６-14　DSP_SIO out



- Number of output Channels (1 to 255)
  送るデータのチャンネル数を設定します。データはここで設定した幅のベクトルをこのブロックみ入力します。

ＳＩＯ（DSP）入力デバイスドライバとＳＩＯ（DSP）出力デバイスドライバは、２台のＤＳＰの間でＳＩＯポートを使用してデータを非同期に交換します。ＳＩＯポートは１つしか有りませんが、これを使い複数のデータを交換できる様に、データを識別する論理チャンネル番号を想定しています。入力（出力）デバイスドライバに入出力するデータはベクトルとし、この各要素をチャンネル番号と１対１に対応付けています。従って、送受信双方のブロックのチャンネル数は、同じでないといけません。

２．６．１７．ＳＩＯ(DSP)初期化デバイスドライバーブロック

ＳＩＯ(DSP)入力（出力）デバイスドライバーブロックを使用する際は、このブロックを使い、ＳＩＯポートの初期化が必要です。
このブロックは、ＳＩＯ(RS232C)初期化ブロックと一緒に使う事はできません。

図 ２．６-15　SIO_DSP Init



- SIO Bit Rate (Bit / Sec)
  ＳＩＯポートの転送速度（ボーレート）を設定します。通信相手のＤＳＰの同ブロックの設定と同じでないといけません。

２．６．１８．ＤＳＰ間ＤＰ通信アービトレーション

ＤＳＰ間のＤＰＲＡM通信デバイスブロックによる交信は、データの同時性（ブロックから取り出したデータ（送ったデータ）が全て同一時刻である事）を保証しています。つまり、「データの前半が後半より１時刻（ステップサイズ）分、昔のデータであった」等ということが無いように、送る側と受ける側でフラグを用いたソフトウェア同期処理を行っています。
これは、複数のＤＳＰが異なる周期で計算を行う、非同期処理間でデータを交換する為には必要な機構です。

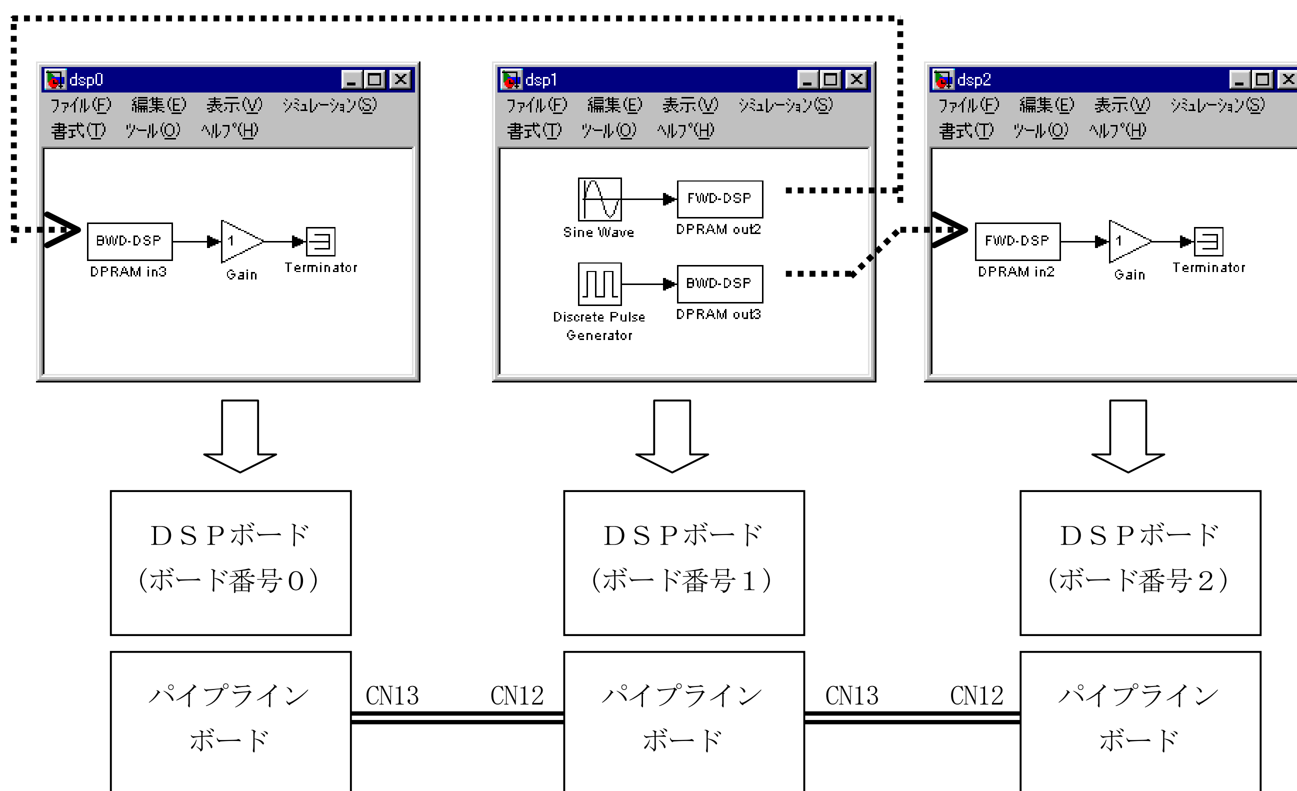
本ソフトウェアでは、データの同時性は、以下の考え方によりアービトレーションしています。
１、書き込み側の要求が先に発生した場合
　読み出し側は書き込み側が完了するまで待機して後読み出す。
２、読み出し側の要求が先に発生した場合
　書き込み側は読み出しみ側が完了するまで待機して後書き込む。
３、両方の要求が同時に発生した場合
　読み出し側は書き込み側が完了するまで待機して後読み出す。

上記３項は、同時の場合は、最新のデータを渡す為、読み出し側が待機します。

２．６．１９．ＤＳＰの前段と後段の意味

ＤＳＰ間のＤＰＲAM通信デバイスブロックには、前段（ＦWD）と後段（ＢWD）があります。前段とは当該ＤＳＰボードに接続されているパイプラインボード（ADSP324-07A 等）のコネクタＣＮ１２の先に接続されたＤＳＰボードであり､後段とは同ＣＮ１３の先に接続されているＤＳＰボードの事を表します。つまり、前段・後段の関係は、パイプラインボードの接続がどうなっているかにより決定され、個々のＤＳＰボードのボード番号には全く関係ありません。

以下に簡単な例を示します。説明の便宜上各ＤＳＰにはボード番号０、１、２と番号を付けたとします。下の図では、真中のＤＳＰボード（ボード番号１）から見て前段とは、同ボードに接続されたパイプラインボードのＣＮ１２に接続されたＤＳＰボードですから、ボード番号０が前段になり、同様に後段は、同ボードに接続されたパイプラインボードのＣＮ１３に接続されたＤＳＰボードですから、ボード番号２が後段になります。図のブロック線図を結ぶ破線矢印 ···> は信号の流れを表しています。



２．６．２０．外部ＣＰＵ（ECU）とのデータ対応

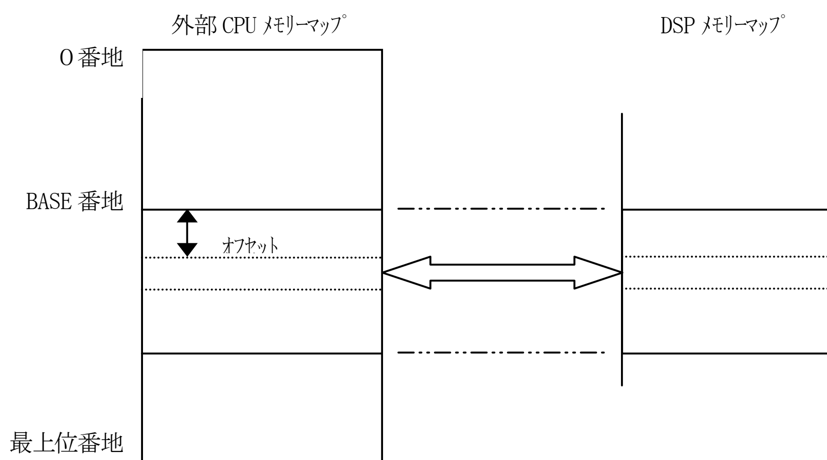
この項では､ＥＣＵ入力／出力ブロックのデータ型と外部ＣＰＵから見たメモリーマップとの対応を説明します。
初期化ダイアログボックスに設定可能なパラメータには以下に示す制限があります。

表　２．６．1　設定可能なパラメータの組み合わせ

データ型 (Data type)		データバイト幅（Data byte size）		
		1	2	4
1	ビット（bit）	可能	可能	不可
2	無符号整数（unsigned int）	可能	可能	制約１
3	符号付き整数（int）	可能	可能	制約２

制約１、７ＦＦＦＦＦＦＦＨより大きい値は与えられません。又、有効桁数が２の２４乗より大きい値を与えた場合、データの有効桁数が２の２４乗以下の部分は切り捨てられます。
制約２、有効桁数が２の２４乗より大きい値を与えた場合、データの有効桁数が２の２４乗以下の部分は切り捨てられます。

外部ＣＰＵはデータバス幅が１６ビットである事を前提としています。
外部ＣＰＵのメモリー番地はバイトアドレスで表示しています。
以下の説明では、外部ＣＰＵのベースアドレスをＢＡＳＥと表記しています。このアドレスは、Ｉ／Ｆハードウェアにより異なります。詳しくはＩ／Ｆハードウェアの説明書を参照してください。



ビットデータ（Data type １）　バイト幅（Byte size）１

表 ２.６.２ データチャンネル番号（ビットデーター１バイト）

アドレス	ビット位置							
	D7	D6	D5	D4	D3	D2	D1	D0
BASE + ｵﾌｾｯﾄ ＋０	7	6	5	4	3	2	1	0
＋１	－	－	－	－	－	－	－	－
＋２	１５	１４	１３	１２	１１	１０	9	8
＋３	－	－	－	－	－	－	－	－
	(以下同様)							

ブロックライブラリへの入出力ベクトルの、先頭の要素がチャンネル番号０のデータに対応します。

ビットデータ（Data type １）　バイト幅（Byte size）２

表 ２.６.３ データチャンネル番号（ビットデーター２バイト）

アドレス	ビット位置							
	D7	D6	D5	D4	D3	D2	D1	D0
BASE + ｵﾌｾｯﾄ ＋０	7	6	5	4	3	2	1	0
＋１	１５	１４	１３	１２	１１	１０	9	8
＋２	２３	２２	２１	２０	１９	１８	１７	１６
＋３	３１	３０	２９	２８	２７	２６	２５	２４
	(以下同様)							

ブロックライブラリへの入出力ベクトルの、先頭の要素がチャンネル番号０のデータに対応します。

無符号整数（Data type ２）　バイト幅（Byte size）１

符号付き整数（Data type ３）　バイト幅（Byte size）１

表 ２．６．４ データチャンネル番号（整数－１バイト）

アドレス	ビット位置							
	D7	D6	D5	D4	D3	D2	D1	D0
BASE + ｵﾌｾｯﾄ ＋０	0-D7	0-D6	0-D5	0-D4	0-D3	0-D2	0-D1	0-D0
＋１	－	－	－	－	－	－	－	－
＋２	1-D7	1-D6	1-D5	1-D4	1-D3	1-D2	1-D1	1-D0
＋３	－	－	－	－	－	－	－	－
	(以下同様)							

表の Ｍ－Ｄｘは、データチャンネル番号Mのビットａｘを表しています。
ブロックライブラリへの入出力ベクトルの、先頭の要素がチャンネル番号０のデータに対応します。
無符号整数は、データ値範囲は０～２５５(FFH)です。
符号付き整数はＤ７が符号ビットで、データ値範囲は－１２８(80H)～１２７（7FH）です。

無符号整数（Data type ２）　バイト幅（Byte size）２

符号付き整数（Data type ３）　バイト幅（Byte size）２

表 ２．６．５ データチャンネル番号（整数－２バイト）

アドレス	ビット位置							
	D7	D6	D5	D4	D3	D2	D1	D0
BASE + ｵﾌｾｯﾄ ＋０	0-D7	0-D6	0-D5	0-D4	0-D3	0-D2	0-D1	0-D0
＋１	0-D15	0-D14	0-D13	0-D12	0-D11	0-D10	0-D9	0-D8
＋２	1-D7	1-D6	1-D5	1-D4	1-D3	1-D2	1-D1	1-D0
＋３	1-D15	1-D14	1-D13	1-D12	1-D11	1-D10	1-D9	1-D8
	(以下同様)							

表の Ｍ－Ｄｘは、データチャンネル番号Mのビットａｘを表しています。
ブロックライブラリへの入出力ベクトルの、先頭の要素がチャンネル番号０のデータに対応します。
無符号整数は、データ範囲が０～６５５３５（FFFFH）です。
符号付き整数はＤ１５が符号ビット、データ値範囲は、－３２７６８（8000H）～３２７６７（7FFFH）です。

無符号整数（Data type ２）　バイト幅（Byte size）４

符号付き整数（Data type ３）　バイト幅（Byte size）４

アドレス	ビット位置							
	D7	D6	D5	D4	D3	D2	D1	D0
BASE + ｵﾌｾｯﾄ ＋０	0-D7	0-D6	0-D5	0-D4	0-D3	0-D2	0-D1	0-D0
＋１	0-D15	0-D14	0-D13	0-D12	0-D11	0-D10	0-D9	0-D8
＋２	(書き込み：無効、読み出し：不定)							
＋３	(書き込み：無効、読み出し：不定)							
＋４	0-D23	0-D22	0-D21	0-D20	0-D19	0-D18	0-D17	0-D16
＋５	0-D31	0-D30	0-D29	0-D28	0-D27	0-D26	0-D25	0-D24
＋６	(書き込み：無効、読み出し：不定)							
＋７	(書き込み：無効、読み出し：不定)							
＋８	1-D7	1-D6	1-D5	1-D4	1-D3	1-D2	1-D1	1-D0
＋９	1-D15	1-D14	1-D13	1-D12	1-D11	1-D10	1-D9	1-D8
＋１０	(書き込み：無効、読み出し：不定)							
＋１１	(書き込み：無効、読み出し：不定)							
＋１２	1-D23	1-D22	1-D21	1-D20	1-D19	1-D18	1-D17	1-D16
＋１３	1-D31	1-D30	1-D29	1-D28	1-D27	1-D26	1-D25	1-D24
＋１４	(書き込み：無効、読み出し：不定)							
＋１５	(書き込み：無効、読み出し：不定)							
	(以下同様)							

表の Ｍ－Ｄｘは、データチャンネル番号MのビットＤｘを表しています。

ブロックライブラリへの入出力ベクトルの、先頭の要素がチャンネル番号０のデータに対応します。

無符号データは、Ｄ３１は常に０としてください。データ値範囲は、０～２１４７４８３６４７（7FFFFFFFH）です。

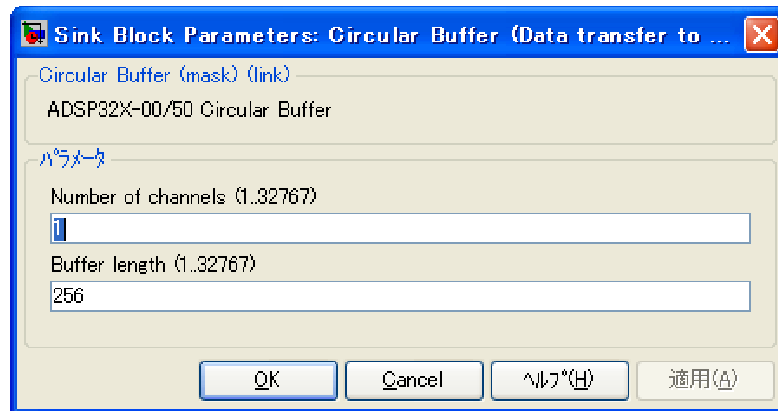
整数データはＤ３１ビットが符号ビットです。データ値範囲は、－２１４７４８３６４８（80000000H）～２１４７４８３６４７（7FFFFFFFH）です。

有効ビット数が２４ビットを超える場合、ＬＳＢ側のビット情報は失われます。これはＤＳＰが取り扱える実数データが、単精度実数（仮数部２４ビット）であることによりますので、回避は出来ません。

２．６．２１．サーキュラーバッファ・デバイスドライバブロック

ＤＳＰのメモリー内に高速にデータ保管が行えるサーキュラーバッファを構築し、ここにデータを貯える為のブロックです。貯えたデータはＵＩＬ（オプション）にて取り出します。

図　２．６-16　C Buffer



- Number of Channels (1 .. 32767)
  このブロックへの入力データの点数を指定します。入力データはベクトルで与えますので、このベクトルの幅と同じです。
- Buffer length (1 .. 32767)
  各データの保存する長さを指定します。

データバッファはＤＳＰ内のメモリーに確保されます。その概ねの長さは、

バッファ長さ　＝　（データ点数＋１）×（データの長さ＋α）

αはバッファの長さの最小単位が２のＮ乗である事から、その最小単位に切り上げる為の係数です。この領域は、実行時にヒープ領域から確保されますので、充分な長さのヒープ領域が必要です。
バッファが確保出来たかどうかは、ＵＩＬで確認できます。バッファが確保出来なかった場合は、データの保管は行われません。その場合、データ保管する長さを減らす等の再試行は、ブロック線図の変更とＲＴＷの再実行により行ってください。

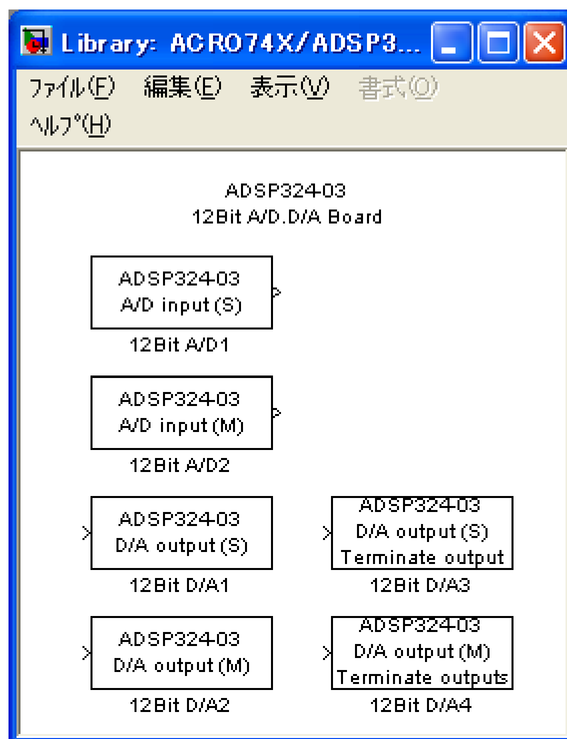
バッファは、サーキュラーバッファと呼ばれる、輪状につながったエンドレスのバッファを構築しています。バッファ内のデータが満杯になった段階で、新たなデータは最古のデータに上書きされ、最新の Buffer length 点のデータが、バッファに残ります。
データはＵＩＬにて取り出します。現在何点のデータが格納されているか等の情報は、ＵＩＬで取り出すことができます。

２．７．　ADSP324-03 デバイスドライバブロック

２．７．１．　概要

図　２．７－１に、ADSP324-03(１２ビットA/D･D/Aボード)用デバイスドライバウィンドウを開いた所を示します。

図　２．７－１　ADSP324-03 ドライバウィンドウ



この中には ADSP324-03 とのインターフェースを取るための４つのデバイスドライバブロックが登録されています。

- A/D input (S)　１チャンネルの A/D 入力デバイスドライバブロックです。
- A/D input (M)　多チャンネルの A/D 入力デバイスドライバブロックです。
- D/A input (S)　１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M)　多チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (S) Terminate output
  終了時の出力指定が行える、１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時の出力指定が行える、多チャンネルの D/A 出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

２．７．２．　チャンネル番号とボードとの対応

ADSP324-03 は最大４台使用でき、１台のボードにA/D、D/Aが各４チャンネル実装されています。ADSP324-03 の各デバイスドライバブロックはこれらのチャンネルを０～１５の連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 ２．７-１の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ (5ページ) を参照してください。

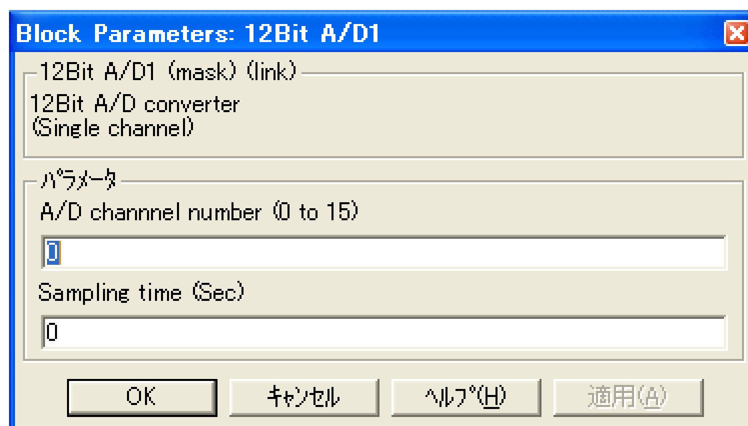
表 ２．７-１　チャンネル番号とＡＤＳＰ３２４－０３ボードとの対応

ボード	IN/OUT0	IN/OUT1	IN/OUT2	IN/OUT3
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

２．７．３．　１チャンネル A/D 入力デバイスドライバブロック

１チャンネル A/D 入力デバイスドライバブロックは、指定されたチャンネル番号の A/D からデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

図　２．７－２　ADSP324-03 A/D input （S）



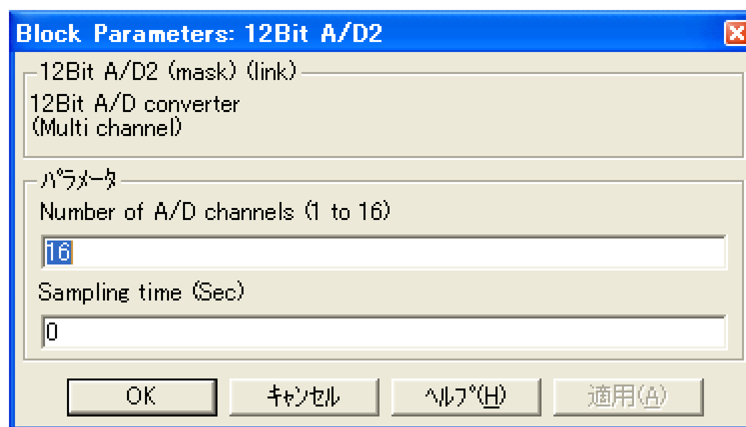
- A/D channel number （0 to 15）
  入力するＡ／Ｄのチャンネル番号を０～１５の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．７－１　(59ページ)　を参照してください。

- Sampling time （Sec）
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

２．７．４．　多チャンネル A/D 入力デバイスドライバブロック

　多チャンネル A/D 入力デバイスドライバブロックは、複数チャンネルの A/D から一括してデータを入力します。入力の対象となる A/D チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

図　２．７－３　ADSP324-03 A/D input (M)



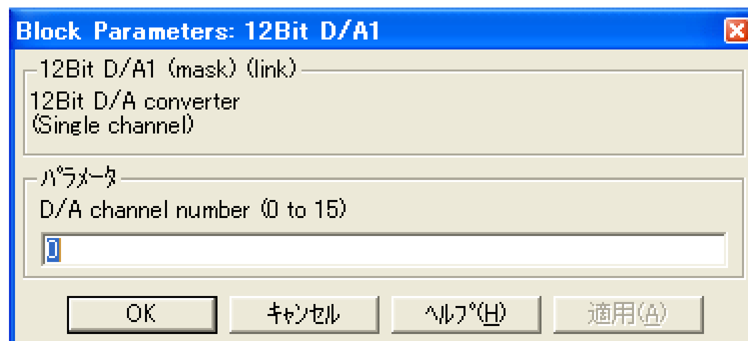
- Number of A/D channels (1 to 16)
  入力するA/Dのチャンネル数を１～１６の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．７－１ (59ページ) を参照してください。

- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

２．７．５．　１チャンネル D/A 出力デバイスドライバブロック

　１チャンネル D/A 出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．７－４　ADSP324-03 D/A output　(S)

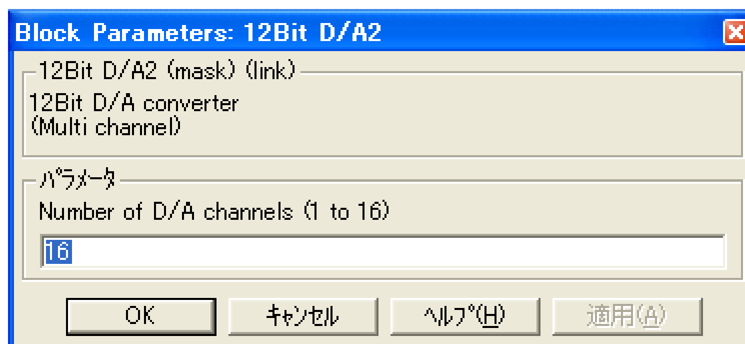


- D/A channel number (0 to 15)
  出力するD/Aのチャンネル番号を０～１５の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．７－１ (59ページ) を参照してください。

２．７．６．　多チャンネル D/A 出力デバイスドライバブロック

多チャンネル D/A 出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。出力の対象となる D/A チャンネルは、チャンネル番号０から連続するチャンネルで､先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０の D/A へ出力されます。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．７－５　ADSP324-03 D/A output (M)



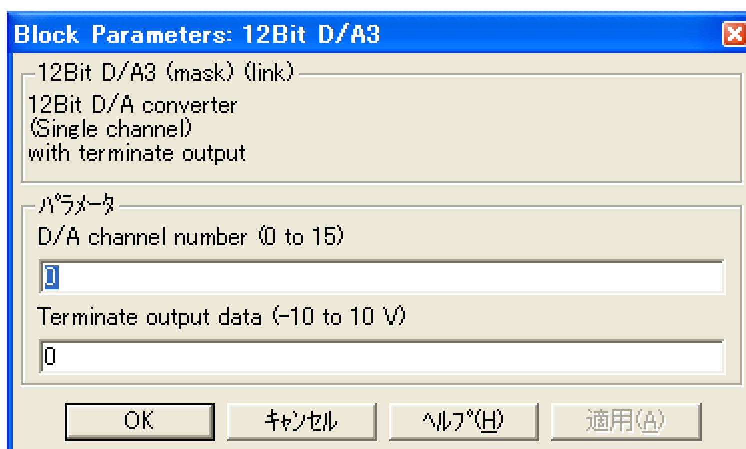
- Number of A/D channels (1 to 16)
  出力するD/Aのチャンネル数を１～１６の整数で指定します｡チャンネル番号とハードウェアとの対応は、表 ２．７－１ (59ページ) を参照してください。

２．７．７．　終了時出力指定１チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き１チャンネル D/A 出力デバイスドライバブロックは､スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。

このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．７－６　ADSP324-03 D/A output (S) Terminate output



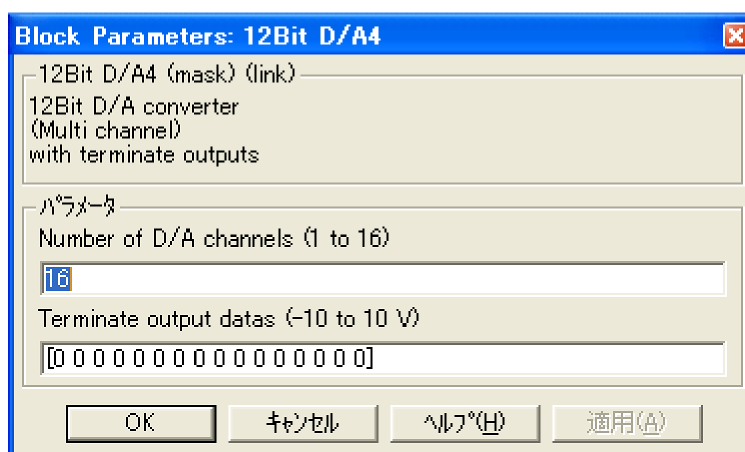
- D/A channel number (0 to 15)
  出力するD/Aのチャンネル番号を０～１５の整数で指定します｡チャンネル番号とハードウェアとの対応は、表　２．７－１ (59ページ) を参照してください。

- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際に D/A から出力する値を設定します。単位は[Volt]です。

２．７．８．　終了時出力指定 多チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き多チャンネル D/A 出力デバイスドライバブロックは､ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０の D/A へ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[V]です｡物理単位からの変換は Gain ブロック等を使って予め行ってください｡

図　２．７－７　ADSP324-03 D/A output (M) Terminate output



- Number of D/A channels (1 to 16)
  出力するD/Aのチャンネル数を１～１６の整数で指定します｡先頭チャンネル番号は０で固定です｡チャンネル番号とハードウェアとの対応は、表　２．７－１ (59ページ) を参照してください。

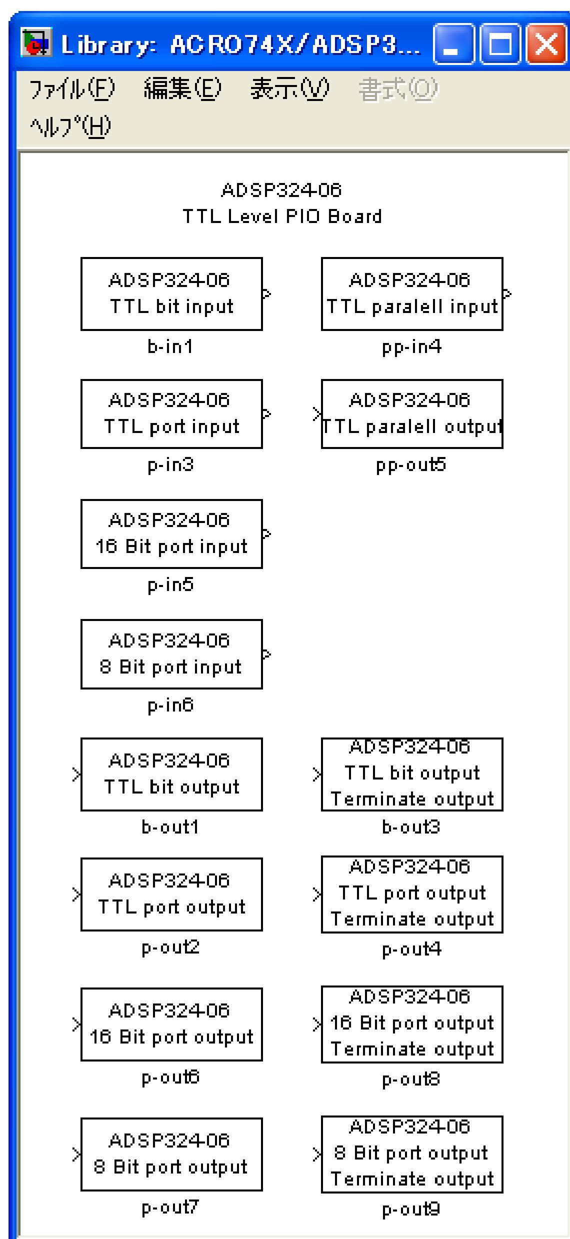
- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際に D/A から出力する値をベクトルで設定します。単位は[Volt]です。

２．８．　ADSP324-06 デバイスドライバブロック

２．８．１．　概要

図 ２．８−１に、ADSP324-06(TTL レベルPIOボード)用デバイスドライバウィンドウを開いた所を示します。

図 ２．８−１　ADSP324-06 ドライバウィンドウ



この中には、ADSP324-06 とのインターフェースを取るための１４個のデバイスドライバブロックが登録されています。

- TTL bit input　１ビット単位の入力デバイスドライバです。
- TTL port input　ポート単位の入力デバイスドライバです。
- TTL 16Bit input　１６ビット単位の入力デバイスドライバです。
- TTL 8Bit input　８ビット単位の入力デバイスドライバです。
- TTL bit output　ビット単位の出力デバイスドライバです。
- TTL port output　ポート単位の出力デバイスドライバです。
- TTL 16Bit output　１６ビット単位の出力デバイスドライバです。
- TTL 8Bit output　８ビット単位の出力デバイスドライバです。
- TTL paralell input　複数ビットの同時入力デバイスドライバです。
- TTL paralell output　複数ビットの同時出力デバイスドライバです。
- TTL bit output Terminate output
  終了時の出力指定が行える、ビット単位の出力デバイスドライバです。
- TTL port output Terminate output
  終了時の出力指定が行える、ポート単位の出力デバイスドライバです。
- TTL 16Bit output Terminate output
  終了時の出力指定が行える、１６ビット単位の出力デバイスドライバです。
- TTL 8Bit output Terminate output
  終了時の出力指定が行える、８ビット単位の出力デバイスドライバです。

ビット単位のデバイスドライバは､任意のポートの､任意のビットをランダムにアクセスできます。外部機器で２値化されたデータ、操作スイッチ、リミットセンサー等のデジタル信号を取り扱うことができます。

ポート単位のデバイスドライバは、ポートを３２ビット符号付き整数として取り扱います。データ形式はストレートバイナリで、-2^{31} ～ $2^{31}-1$ の範囲のデータとなります。外部機器でデジタル化されたデータを取り扱うことができます。

複数ビットのデバイスドライバは、複数のビットを同時に入出力します。ビット単位のデバイスドライバを複数並べ、Mux (Demux) にてベクトル／スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバの方がビット単位のデバイスドライバを複数並べるより高速に入出力が行えます。

終了時出力指定の行えるデバイスドライバは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

２．８．２．　ポート／ビット番号とボードとの対応

ADSP324-06 は最大４台使用できます。ADSP324-06 にはI/Oポートが１ポートあり、各デバイスドライバブロックはこれらポートに論理ポート番号（論理ビット番号）を割り付けて取り扱います。ADSP324-06 の各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表 ２．８－１　及び　表 ２．８－２　の通りです。
ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表 ２．８－１　ポート番号と ADSP324-06 ボードとの対応

ボード	入出力信号名	ポート番号		
		８ビット単位	１６ビット単位	ポート単位
１台目	ID/OD00～07	0	0	0
	ID/OD08～15	1		
	ID/OD16～23	2	1	
	ID/OD24～31	3		
２台目	ID/OD00～07	4	2	1
	ID/OD08～15	5		
	ID/OD16～23	6	3	
	ID/OD24～31	7		
３台目	ID/OD00～07	8	4	2
	ID/OD08～15	9		
	ID/OD16～23	１０	5	
	ID/OD24～31	１１		
４台目	ID/OD00～07	１２	6	3
	ID/OD08～15	１３		
	ID/OD16～23	１４	7	
	ID/OD24～31	１５		

表　２．８−２　ビット番号と ADSP324-06 ボードとの対応

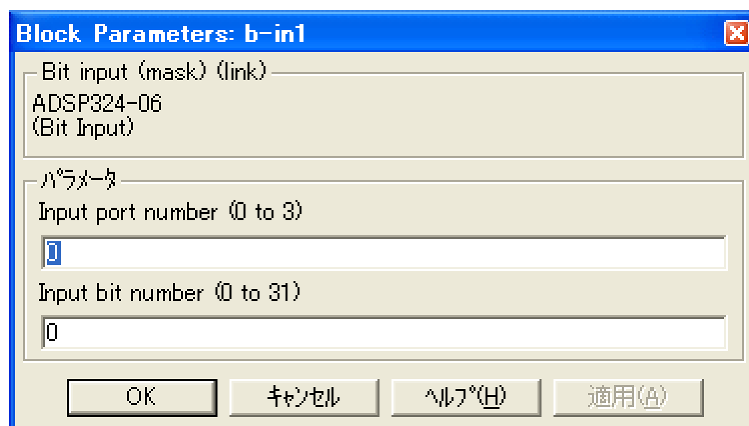
入出力信号名	ビット番号／ポート単位			
	１ビット単位	８ビット単位	１６ビット単位	ポート単位
ID/OD31	３１	符号	符号	符号
ID/OD30	３０	2^{6}	2^{14}	2^{30}
ID/OD29	２９	2^{5}	2^{13}	2^{29}
ID/OD28	２８	2^{4}	2^{12}	2^{28}
ID/OD27	２７	2^{3}	2^{11}	2^{27}
ID/OD26	２６	2^{2}	2^{10}	2^{26}
ID/OD25	２５	2^{1}	2^{9}	2^{25}
ID/OD24	２４	2^{0}	2^{8}	2^{24}
ID/OD23	２３	符号	2^{7}	2^{23}
ID/OD22	２２	2^{6}	2^{6}	2^{22}
ID/OD21	２１	2^{5}	2^{5}	2^{21}
ID/OD20	２０	2^{4}	2^{4}	2^{20}
ID/OD19	１９	2^{3}	2^{3}	2^{19}
ID/OD18	１８	2^{2}	2^{2}	2^{18}
ID/OD17	１７	2^{1}	2^{1}	2^{17}
ID/OD16	１６	2^{0}	2^{0}	2^{16}
ID/OD15	１５	符号	符号	2^{15}
ID/OD14	１４	2^{6}	2^{14}	2^{14}
ID/OD13	１３	2^{5}	2^{13}	2^{13}
ID/OD12	１２	2^{4}	2^{12}	2^{12}
ID/OD11	１１	2^{3}	2^{11}	2^{11}
ID/OD10	１０	2^{2}	2^{10}	2^{10}
ID/OD09	9	2^{1}	2^{9}	2^{9}
ID/OD08	8	2^{0}	2^{8}	2^{8}
ID/OD07	7	符号	2^{7}	2^{7}
ID/OD06	6	2^{6}	2^{6}	2^{6}
ID/OD05	5	2^{5}	2^{5}	2^{5}
ID/OD04	4	2^{4}	2^{4}	2^{4}
ID/OD03	3	2^{3}	2^{3}	2^{3}
ID/OD02	2	2^{2}	2^{2}	2^{2}
ID/OD01	1	2^{1}	2^{1}	2^{1}
ID/OD00	0	2^{0}	2^{0}	2^{0}

注：各ボード共通

２．８．３．　ビット単位入力デバイスドライバブロック

ビット単位入力デバイスドライバブロックは、指定されたポートの指定されたビットを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードへの入力レベルが LOW の時０、HIGH の時１です。入力信号が接続されていないビットは１になります。ボードが実装されていないポートの全てのビットは不定です。

図 ２．８−２　ADSP324-06 TTL bit input



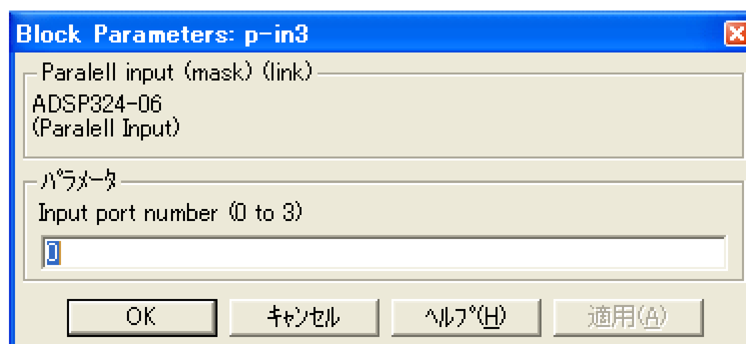
- Input port number (0 to 3)
  入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８−１ (68ページ) を参照してください。

- Input bit number (0 to 31)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．８−２ (69ページ) を参照してください。

２．８．４．　ポート単位入力デバイスドライバブロック

ポート単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１とした、符号付き３２ビット整数とみなした値です。
ボードが実装されていないポートの値は不定です。

図　２．８−３　ADSP324-06 TTL port input

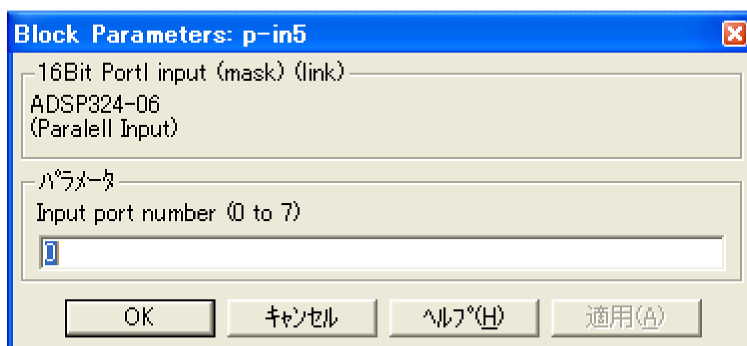


- Input port number (0 to 3)
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．８−１ (68ページ) を参照してください。

２．８．５．　１６Bit port 単位入力デバイスドライバブロック

　１６ビット単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１とした、符号付き１６ビット整数とみなした値です。
入力が接続されていないポート及びボードが実装されていないポートの値は不定です。

図 ２．８－４　ADSP324-06 １６Bit port input

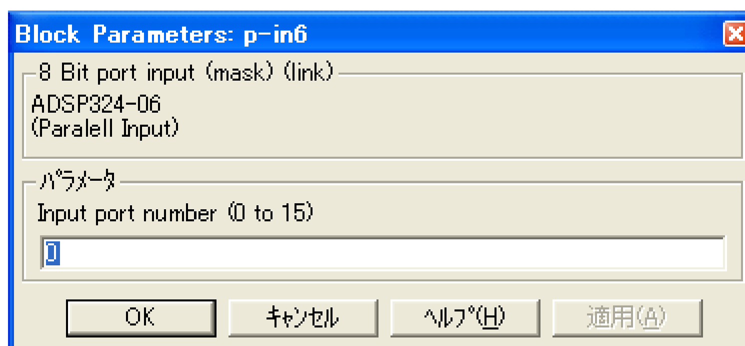


- Input port number (0 to 7)
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．８－１ (68ページ) を参照してください。

２．８．６．　８Bit port 単位入力デバイスドライバブロック

８ビット単位入力デバイスドライバブロックは、指定されたポートを入力し、結果をスカラーで出力します。このブロックから出力される値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１とした、符号付き８ビット整数とみなした値です。
入力が接続されていないポート及びボードが実装されていないポートの値は不定です。

図 ２．８－５　ADSP324-06 ８Bit port input

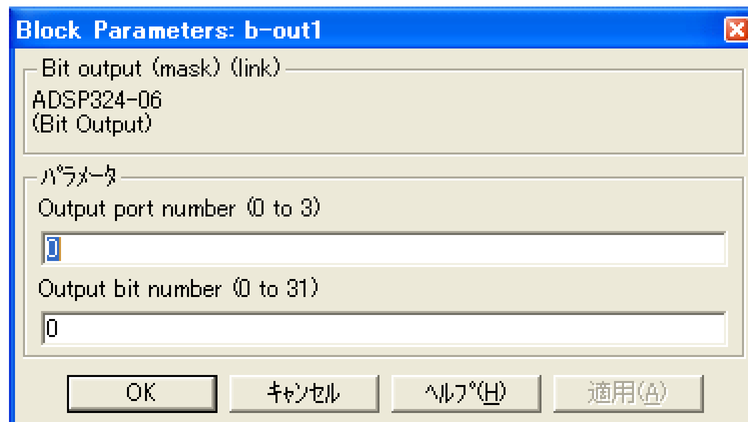


- Input port number (0 to 15)
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．８－１ (68ページ) を参照してください。

２．８．７．　ビット単位出力デバイスドライバブロック

ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。

図 ２．８－６　ADSP324-06 TTL bit output



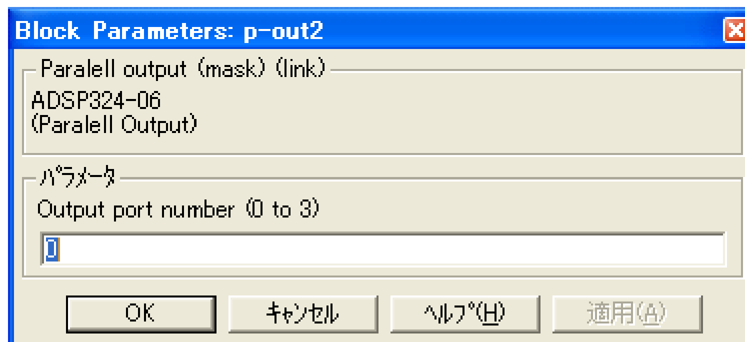
- Output port number (0 to 3)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１ (68ページ) を参照してください。
- Output bit number (0 to 31)
  出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．８－２ (69ページ) を参照してください。

注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

２．８．８．　ポート単位出力デバイスドライバブロック

ポート単位出力デバイスドライバブロックは、スカラーの入力を受取り、符号付き３２ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。

図　２．８－７　ADSP324-06 TTL port output



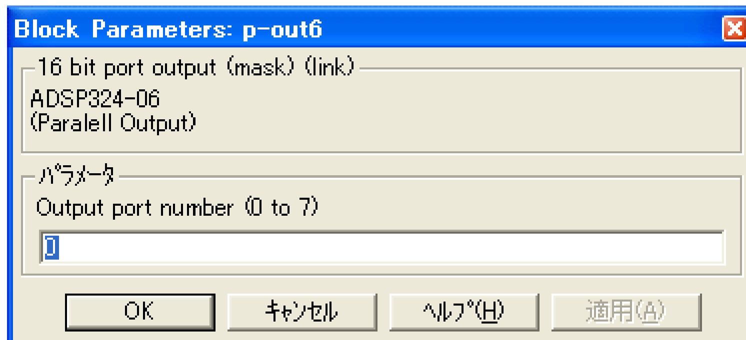
- Output port number (0 to 3)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．８－１（68ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

２．８．９．　１６Bit port 単位出力デバイスドライバブロック

　１６ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き１６ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。このブロックが出力できる値は、-2^{15}～$2^{15}-1$ の範囲です。この範囲を超える値を出力しようとすると、値が取り扱える範囲に収まる様に、＋側は＋の、－側は－の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

図 ２．８－８　ADSP324-06 16 Bit port output



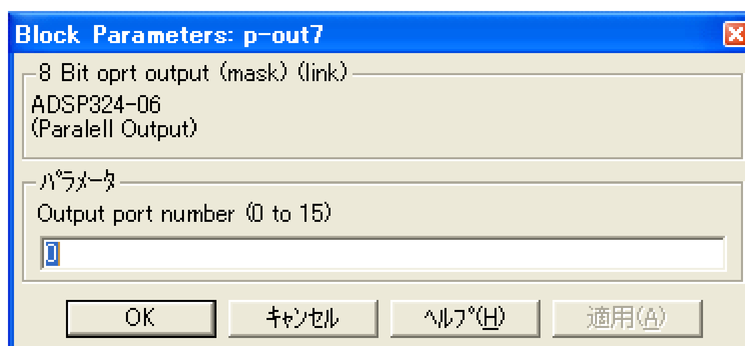
- Output port number (0 to 7)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。

注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

２．８．１０．　８Bit port 単位出力デバイスドライバブロック

８ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き８ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。このブロックが出力できる値は、-2^{7}～$2^{7}-1$ の範囲です。この範囲を超える値を出力しようとすると、値が取り扱える範囲に収まる様に、＋側は＋の、－側は－の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。

図 ２．８－９　ADSP324-06 8 Bit port output



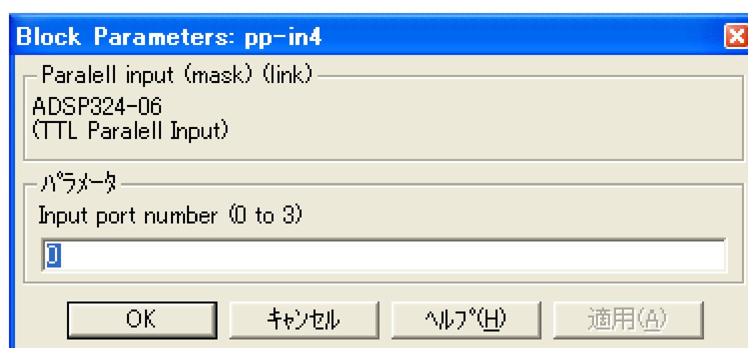
- Output port number (0 to 15)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。

注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

２．８．１１．　複数ビット入力デバイスドライバブロック

複数ビット入力デバイスドライバブロックは、指定されたポートを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは幅３２のベクトルです。ベクトルの先頭の要素がビット番号０、次がビット番号１の順で、最後の要素がビット番号３１に対応します。ベクトルの各要素の値は、ボードの各ビットへの入力レベルが LOW の時０、HIGH の時１です。
ポート入力デバイスドライバはとの相違点は、ポート入力は TTL 入力の１ポートを１つのデータと見なすのに対し、このブロックは１つ１つのビットの状態が個々の信号（０か１）として入力できます。
ボードが実装されていない場合の値は不定です。

図 ２．８－１０　ADSP324-06 TTL paralell input

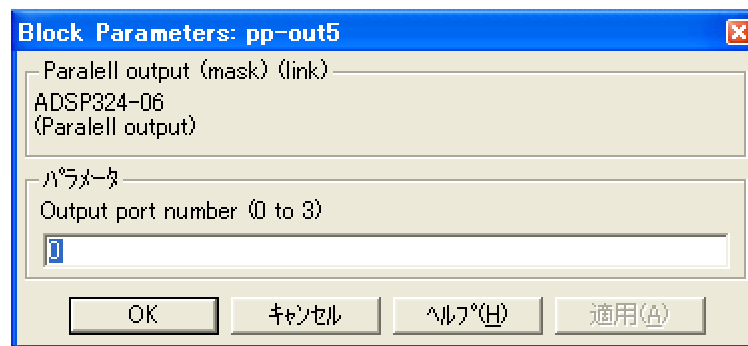


- Input port number (0 to 3)
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．８－１（68ページ）を参照してください。

２．８．１２．　複数ビット出力デバイスドライバブロック

複数ビット出力デバイスドライバブロックは、ベクトルの入力を受取り、各要素に対応するビットに HIGH 又は LOW を出力します。各要素の値が０の時には LOW を、０以外の時には HIGH を出力します。このブロックへの入力は幅３２のベクトルを与えます。ベクトルの先頭の要素がビット番号０、次がビット番号１の順で、最後の要素がビット番号３１に対応します。
ポート出力デバイスドライバはとの相違点は、ポート出力はそのブロックへの入力をバイナリに変換し、各ビットに対応する TTL ポートのビットに HIGH/LOW を出力しますが、このブロックは入力ベクトルの個々の要素状態が TTL ポートの個々のビットに HIGH/LOW で出力されます。

図 ２．８－１１　ADSP324-06 TTL port output



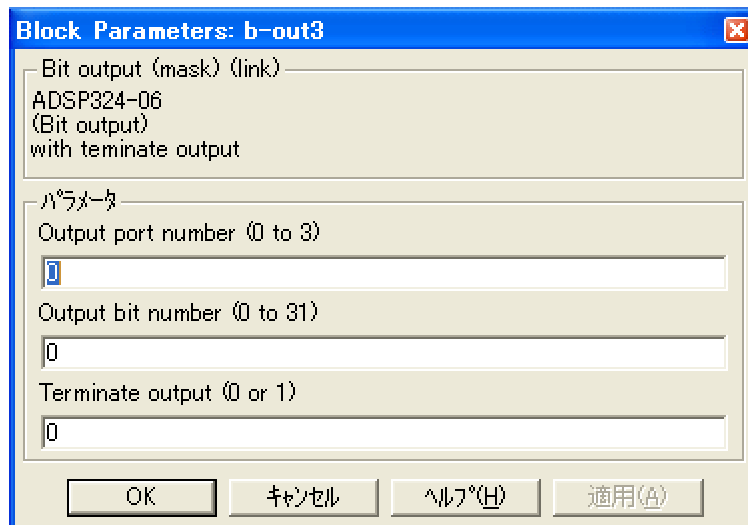
- Output port number (0 to 3)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

２．８．１３．　　終了時出力指定ビット単位出力デバイスドライバブロック

終了時出力指定付きビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号レベルは、このブロックへの入力レベルが０の時に LOW、１の時 HIGH です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 LOW です。
このブロックは実時間モデルが停止する際に、予め設定した値を出力します。

図 ２．８－１２　ADSP324-06 TTL bit output Terminate output



- Output port number (0 to 3)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１ (68ページ) を参照してください。

- Output bit number (0 to 31)
  出力の対象となるビット番号を指定します。ビット番号とハードウェアとの対応は、表 ２．８－２ (69ページ) を参照してください。

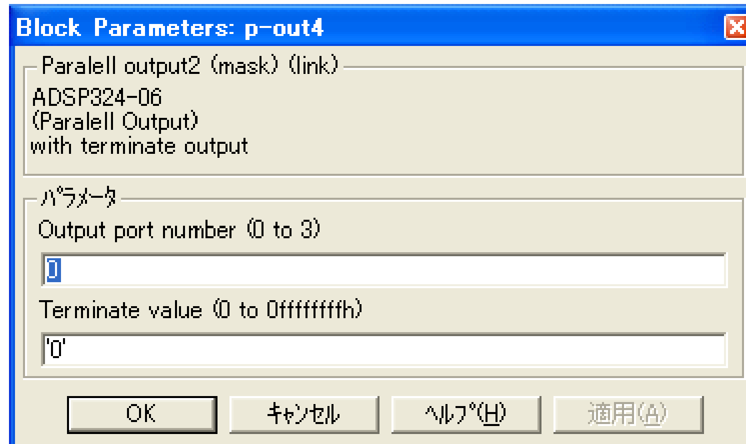
- Output bit number (0 or 1)
  実時間モデルが停止する際に出力して欲しい状態を設定します。０で LOW, １で HIGH が出力されます。

注意：ポート単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ポート単位出力デバイスドライバブロックにより失われます。

２．８．１４．　終了時出力指定ポート単位出力デバイスドライバブロック

終了時出力指定付きポート単位出力デバイスドライバブロックは、スカラーの入力を受取り、符号付き３２ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。
このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 ２．８－１３　ADSP324-06 TTL port output Terminate output



- Output port number （0 to 3）
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。

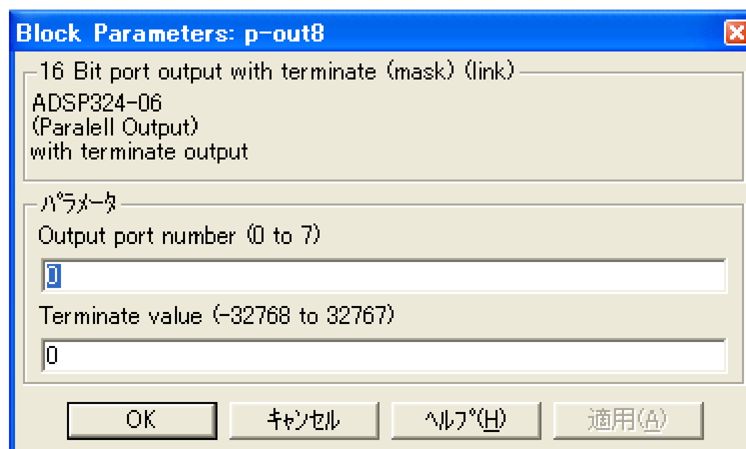
- Terminate value （0 to 0ffffffffh）
  実時間モデルが停止する際に出力して欲しい値を１６進数で設定します。

注意：ビット単位出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は、ビット単位出力デバイスドライバブロックにより失われます。

２．８．１５．　終了時出力指定 １６Bit port 単位出力デバイスドライバブロック

終了時出力指定付き１６ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き１６ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。このブロックが出力できる値は、-2^{15}～$2^{15}-1$ の範囲です。この範囲を超える値を出力しようとすると、値が取り扱える範囲に収まる様に、＋側は＋の、－側は－の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。
このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 ２．８－１４　ADSP324-06 16 Bit port output Terminate output



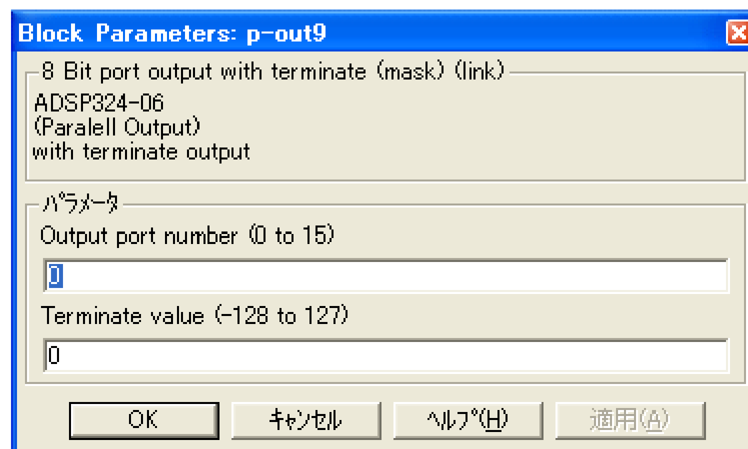
- Output port number (0 to 7)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。
- Terminate value (0 to 0ffffffffh)
  実時間モデルが停止する際に出力して欲しい値を１６進数で設定します。

注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

２．８．１６．　終了時出力指定８Bit port 単位出力デバイスドライバブロック

終了時出力指定付き８ビット単位出力デバイスドライバブロックは、スカラーの入力を受取り、範囲制限を行った上で符号付き８ビット整数に変換した後、指定されたポートへ出力します。ボードから出力される信号レベルは、変換後の整数の各ビットが０の時に LOW、１の時 HIGH です。このブロックが出力できる値は、-2^7～2^7-1 の範囲です。この範囲を超える値を出力しようとすると、値が取り扱える範囲に収まる様に、＋側は＋の、－側は－の最大値（絶対値が最大の値）に制限されます。従って出力データのラップラウンドは発生しません。
このブロックは、実時間モデルが停止する際に予め指定した値を出力します。

図 ２．８－１５　ADSP324-06 8 Bit port output Terminate output



- Output port number (0 to 15)
  出力先ポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．８－１（68ページ）を参照してください。
- Terminate value (0 to 0ffffffffh)
  実時間モデルが停止する際に出力して欲しい値を１６進数で設定します。

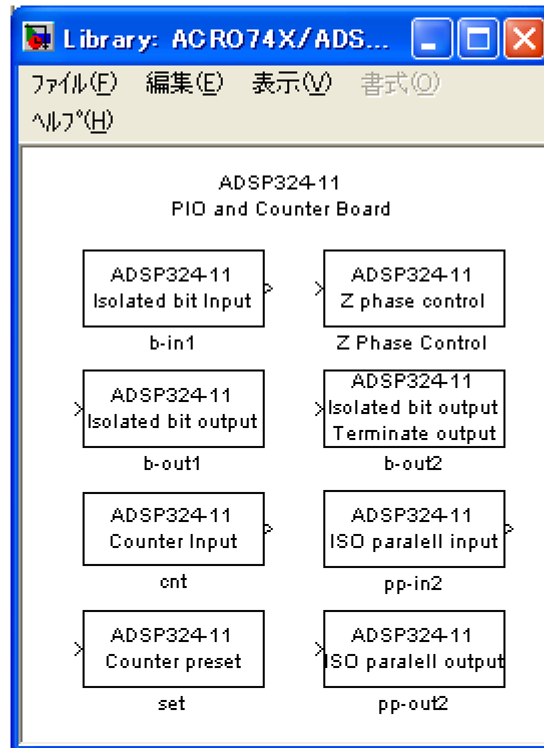
注意：他の出力デバイスドライバブロックとは同じポートを指標できません。同じポートを指標した場合、このブロックで出力した結果は失われ、出力状態は予測不能となります。

２．９．　ADSP324-11 デバイスドライバブロック

２．９．１．　概要

図 ２．９－１に、ADSP324-11(PIO&カウンタボード) 用デバイスドライバーウィンドウを開いた所を示します。

図 ２．９－１　ADSP324-11 ドライバウィンドウ



この中には、ADSP324-11 とのインターフェースを取るためのデバイスドライバブロックが登録されています。

- Isolated bit input　　ビット単位の絶縁入力デバイスドライバブロックです。
- Isolated bit output　ビット単位の絶縁出力デバイスドライバブロックです。
- Counter input　　　　プリセット付きカウンタデバイスドライバです。
- Counter preset　　　カウンタープリセットデバイスドライバブロックです。
- Isolated bit output Teminate output
  終了時の出力指定が行えるビット単位の絶縁出力デバイスドライバブロックです。
- Isolated paralell input
  複数ビット同時の絶縁入力デバイスドライバブロックです。
- Isolated paralell output
  複数ビット同時の絶縁出力デバイスドライバブロックです。

ビット単位の入出力デバイスドライバは、ADSP324-11 ボードの任意の絶縁入出力ポートの、任意のビットに信号を入出力します。アクセスは全てビット単位で行われます。

　プリセット付きカウンタデバイスドライバは、カウンタをアクセスするデバイスドライバで、初期化時にカウンタを規定の値にセットすることもできます。
カウンタプリセットデバイスドライバは、任意のカウンタを、任意の時刻にプリセットするドライバです。

　終了時出力指定の行えるデバイスドライバは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

　複数ビットのデバイスドライバは、複数のビットを同時に入出力します。ビット単位のデバイスドライバを複数並べ、Mux（Demux）にてベクトル／スカラーの変換を行ったのと同等の処理が行えますが、ビットの数が多い場合は、このドライバの方がビット単位のデバイスドライバを複数並べるより高速に入出力が行えます。

２．９．２．　ビット番号、ポート／チャンネル番号とボードとの対応

ADSP324-11 は最大４台使用できます。ADSP324-11 ボードにはカウンタが４チャンネルと絶縁I/Oポートが１ポート実装されています。ADSP324-11 の各デバイスドライバブロックは、これらのポートを０から始まる連続する論理番号で取り扱います。各デバイスドライバブロックが取り扱うポート番号、ビット番号とハードウェアとの対応は、表　２．９－１　及び　表　２．９－２　の通りです。
ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

表　２．９－１　ポート／チャンネル番号と ADSP324-11 との対応

ボード	カウンタチャンネル番号				絶縁ポート番号
	カウンタ０	カウンタ１	カウンタ２	カウンタ３	
１台目	０	１	２	３	０
２台目	４	５	６	７	１
３台目	８	９	１０	１１	２
４台目	１２	１３	１４	１５	３

表　２．９−２　ビット番号と ADSP324-11 との対応

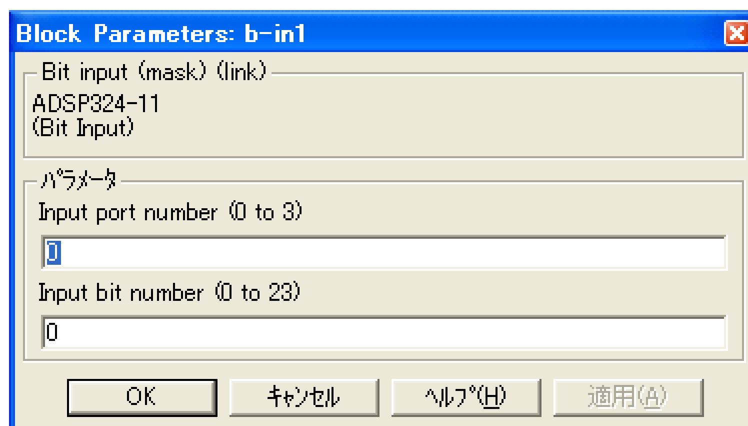
ビット番号	絶縁入力信号名	絶縁出力信号名
０	IO_00	OO_00
１	IO_01	OO_01
２	IO_02	OO_02
３	IO_03	OO_03
４	IO_04	OO_04
５	IO_05	OO_05
６	IO_06	OO_06
７	IO_07	OO_07
８	IO_10	OO_10
９	IO_11	OO_11
１０	IO_12	OO_12
１１	IO_13	OO_13
１２	IO_14	OO_14
１３	IO_15	OO_15
１４	IO_16	OO_16
１５	IO_17	OO_17
１６	IO_20	—
１７	IO_21	—
１８	IO_22	—
１９	IO_23	—
２０	IO_24	—
２１	IO_25	—
２２	IO_26	—
２３	IO_27	—

注１：各絶縁入出力ポート共通

２．９．３．　　ビット単位、絶縁入力デバイスドライバブロック

ビット単位絶縁入力デバイスドライバブロックは､指定されたポートの指定されたビットを入力し､結果をスカラーで出力します｡このブロックから出力される値は､ボードへの入力信号が OFF の時０､ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図　２．９−２　ADSP324-11 Isolated bit input



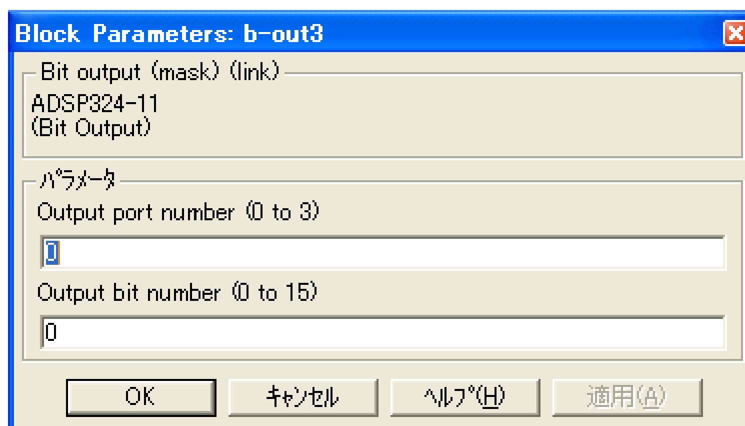
- Input port number (0 to 3)
  入力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．９−１（86ページ）を参照してください。

- Input bit number (0 to 23)
  入力の対象となるビット番号を指定します。指定できるビット番号は０～２３です。ビット番号とハードウェアとの対応は、表　２．９−２（87ページ）を参照してください。

2．9．4．　　ビット単位、絶縁出力デバイスドライバブロック

ビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが０の時に OFF、１の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。

図 ２．９－３　ADSP324-11 Isolated bit output



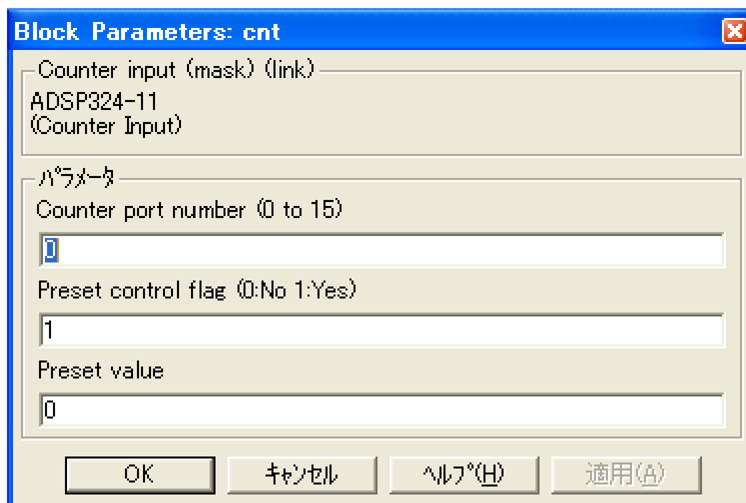
- Output port number (0 to 3)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．９－１ (86ページ) を参照してください。

- Output bit number (0 to 15)
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表 ２．９－２ (87ページ) を参照してください。

２．９．５.　　プリセット付きカウンタデバイスドライバブロック

プリセット付きカウンタデバイスドライバブロックは、任意のチャンネル番号のカウンタ値を読み込みます。このブロックの出力はスカラーです。ADSP324-11 ボードの指定されたカウンタは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行った場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。

図 ２．９－４　ADSP324-11 Counter input



- Counter port number (0 to 15)
  入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表２．９－１（86ページ）を参照してください。

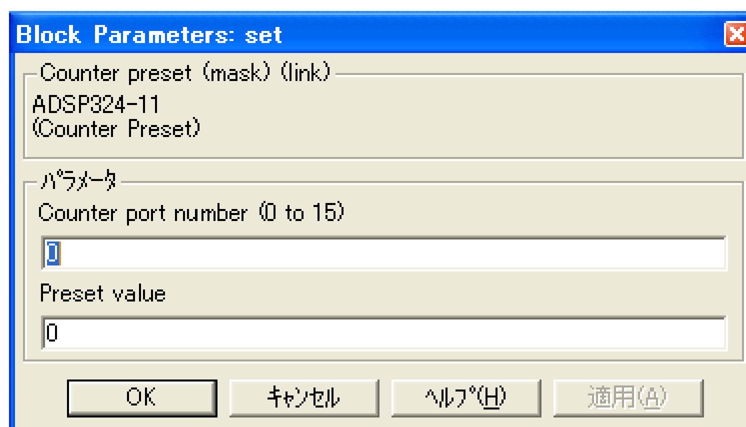
- Preset controll flagg (0:No 1:Yes)
  初期化時にカウンタ値のプリセットを行うか否かを決定します。０又は１を設定してください。０を設定した場合、プリセットは行いません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。

- Preset value
  Preset controll flagg に１を設定した場合、ここに設定した値がカウンタの初期値として採用されます。 Preset controll flagg に１を設定した場合のみ有効です。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

２．９．６．　　カウンタプリセットデバイスドライバブロック

カウンタプリセットデバイスドライバブロックは、任意のチャンネル番号のカウンタを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に１度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。
このブロックには１つのスカラー入力があり、ここに０又は１の値を入力することによりプリセットタイミングを制御します。０が入力されている期間、カウンタはカウント動作を行います。１が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図　２．９－５　ADSP324-11 Counter preset

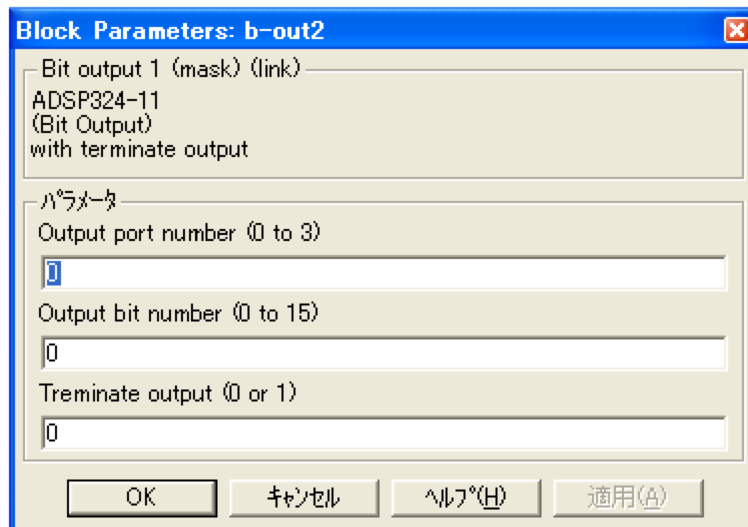


- Counter port number （0 to 15）
  入力の対象となるカウンタポート番号を指定します。ポート番号とハードウェアとの対応は、表２．９－１（86ページ）を参照してください。
- Preset value
  ここに設定した値がカウンタにセットされます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

２．９．７．　　　終了時出力指定ビット単位、絶縁出力デバイスドライバブロック

　終了時出力指定付きビット単位絶縁出力デバイスドライバブロックは、スカラーの入力を受取り、指定されたポートの指定されたビットへ出力します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力レベルが０の時に OFF、１の時 ON です。指定以外のビットの状態は保存されます。ボードの出力の初期値は全ビット共 OFF です。
このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力する事ができます。

図 ２．９－６　ADSP324-11 Isolated bit output Terminate output



- Output port number (0 to 3)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 ２．９－１ (86ページ) を参照してください。

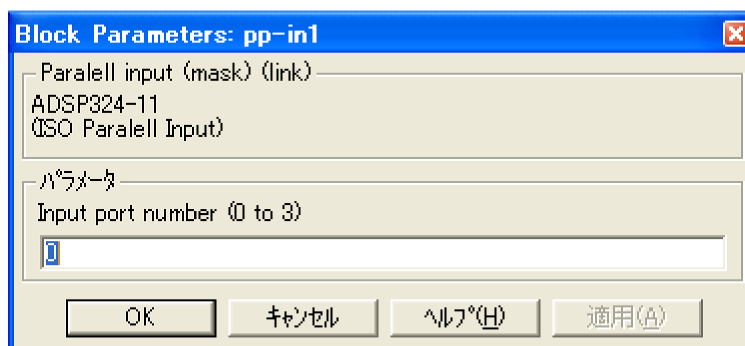
- Output bit number (0 to 15)
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表 ２．９－２ (87ページ) を参照してください。

- Terminale output (0 or 1)
  実時間モデルが終了する際に出力したい値を設定します。

２．９．８.　　複数ビット同時、絶縁入力デバイスドライバブロック

複数ビット同時絶縁入力デバイスドライバブロックは、指定されたポートの全ビットを入力し、結果をベクトルで出力します。このブロックから出力されるベクトルは、幅が２４のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット２３に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 ２．９－７　ADSP324-11 ISO paralell input

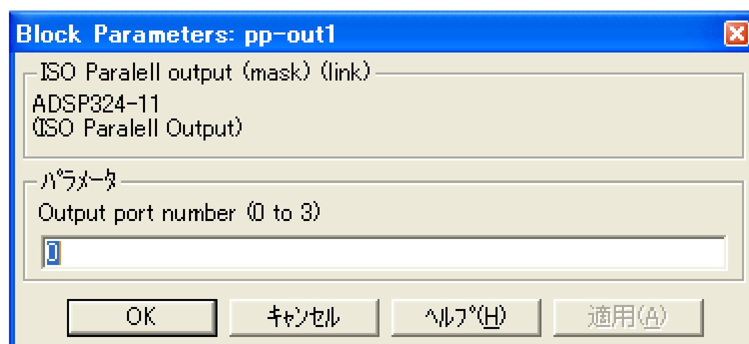


- Input port number (0 to 3)
  入力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．９－１ (86ページ) を参照してください。

２．９．９．　　複数ビット同時、絶縁出力デバイスドライバブロック

複数ビット同時絶縁出力デバイスドライバブロックは、ベクトルの入力を受取り、指定されたポートの各ビットへ出力します。入力するベクトルは幅１６のベクトルで、先頭の要素が出力ビット０、次が出力ビット１の順で、最後の要素が出力ビット１５に相当します。ボードから出力される信号（出力トランジスタ）は、このブロックへの入力が０の時に OFF、０以外の時 ON です。

図 ２．９－８　ADSP324-11 ISO paralell output



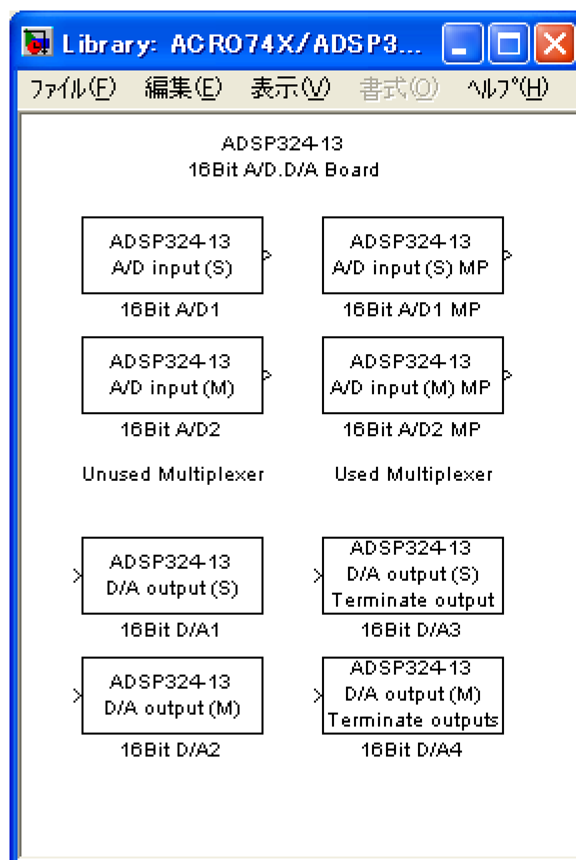
- Output port number (0 to 3)
  出力の対象となるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表２．９－１（86ページ）を参照してください。

２．１０．　ADSP324-13 デバイスドライバブロック

２．１０．１．　　概要

図　２．１０−１に、ADSP324-13(１６ビットA/D・D/Aボード)用デバイスドライバウィンドウを開いた所を示します。

図　２．１０−１　ADSP324-13 ドライバウィンドウ



この中には ADSP324-13 とのインターフェースを取るためのデバイスドライバブロックが登録されています。

- A/D input (S)　　　１チャンネルの A/D 入力デバイスドライバブロックです。
  マルチプレクサ
- A/D input (M)　　　多チャンネデバイスドライバブロックライバブロックです。
  マルチプレクサ
- A/D input (S) MP　１チャンネデバイスドライバブロックライバブロックです。
  マルチプレクサ
- A/D input (M) MP　多チャンネデバイスドライバブロックライバブロックです。
  マルチプレクサ
- D/A input (S)　　　１チャンネルの D/A 出力デバイスドライバブロックです。

- D/A input (M)　　多チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (S)　　Terminate output
  終了時出力指定付きの１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時出力指定付きの多チャンネルの D/A 出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

２．１０．２．　チャンネル番号とボードとの対応

ADSP324-13ボードは最大４台使用できます。ADSP324-13ボードにはA/Dが２チャンネル（マルチプレクサーオプションを増設した場合は１６チャンネル）と、D/Aが２チャンネル実装されています。ADSP324-13の各デバイスドライバブロックは、これらのチャンネルを０から始まる連続する論理チャンネル番号として取り扱います。ADSP324-13の各デバイスドライバブロックが取り扱うチャンネル番号とハードウェアとの対応は、表 ２．１０−１ 及び 表 ２．１０−２の通りです。
ボードの１～４台目の区別についてはハードウェアのセットアップ（5ページ）を参照してください。

チャンネル番号とADSP324-13ボードとの対応

表 ２．１０−１　マルチプレクサが無い場合

ボード	信号名	
	IN/OUT00	IN/OUT01
１台目	0	1
２台目	2	3
３台目	4	5
４台目	6	7

表 ２．１０−２　マルチプレクサが有る場合

ボード	信号名							
	IN00	IN01	…	IN07	IN10	IN11	…	IN17
１台目	0	1	…	7	8	9	…	１５
２台目	１６	１７	…	２３	２４	２５	…	３１
３台目	３２	３３	…	３９	４０	４１	…	４７
４台目	４８	４９	…	５５	５６	５７	…	６３

２．１０．３．　PGA バイパス

マルチプレクサーオプション（ADSP324-12）を追加した場合、A/D 変換は、マルチプレクサを切り替えながら行う為、変換に要する時間が長くなります。これは、A/D チャンネルの切り替えの際に、ボードに実装されている PGA（Programable Gain Amp）や S/H（Sample & Hold）の応答時間（セトリングタイム）を待機している為です。この時間の大半は PGA が占めているため、PGA を取り除く（バイパスさせる）事により大幅に短縮できます。(約３０４μ秒)

PGA をバイパスするには、以下の点の注意が必要です。

（1）　入力がシングルエンドになります。(標準は差動入力です。)

（2）　ハードウェアの改造が必要です。

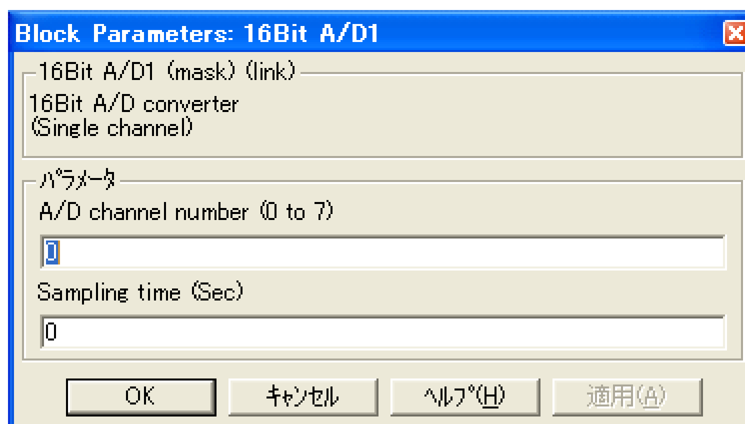
（3）　ハードウェアの改造を行わずに、ブロックの PGA amp パラメータを０にすると、正しい値が取り込めません。

詳細については、メーカーにお問い合わせください。

２．１０．４．　　１チャンネル A/D 入力デバイスドライバブロック

１チャンネル A/D 入力デバイスドライバブロックは、指定されたチャンネル番号の A/D からデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。
このデバイスドライバはマルチプレクサ(オプション)が無い場合のみ利用可能です。

図　２．１０－２　ADSP324-13 A/D input (S)



- A/D channel number (0 to 7)
  入力するA/Dのチャンネル番号を０～７の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１０－１ (97ページ) を参照してください。

- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

２．１０．５．　　多チャンネル A/D 入力デバイスドライバブロック

多チャンネル A/D 入力デバイスドライバブロックは、複数チャンネルの A/D から一括してデータを入力します。入力の対象となる A/D チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。
このデバイスドライバはマルチプレクサ（オプション）が無い場合のみ利用可能です。

図 ２．１０－３　ADSP324-13 A/D input (M)



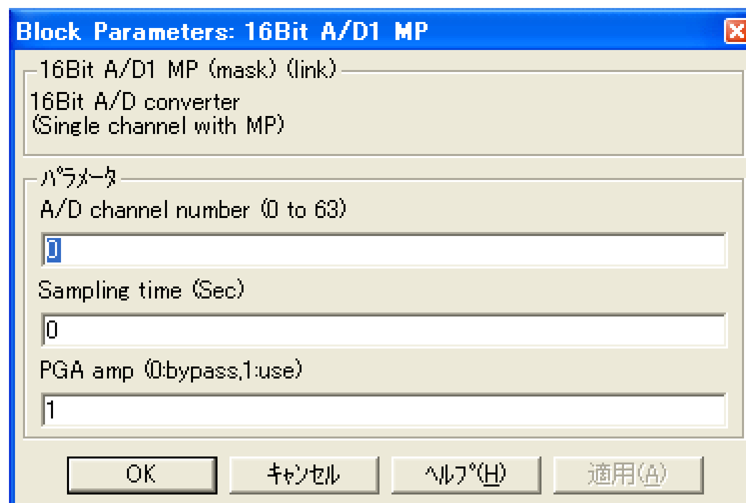
- Number of A/D channels (1 to 8)
  入力するA/Dのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１０－１ (97ページ) を参照してください。

- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

２．１０．６．　１チャンネル A/D 入力(MP 付き)デバイスドライバブロック

１チャンネル A/D 入力（MP＝マルチプレクサ付き）デバイスドライバブロックは、指定されたチャンネル番号の A/D からデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。
このデバイスドライバはマルチプレクサ（オプション）が有る場合のみ利用可能です。

図　２．１０－４　ADSP324-13 A/D input （S）　MP



- A/D channel number （0 to 63）
  入力するA/Dのチャンネル番号を０～６３の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１０－２（97ページ）を参照してください。

- Sampling time （Sec）
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

- PGA amp （0:bypass,1:use）
  ボード上のPGAを使用しているか否を設定します。通常は１を設定してください。AD変換の高速化の為、PGAのバイパス（２．１０．３参照）を実施した場合のみ０としてください。

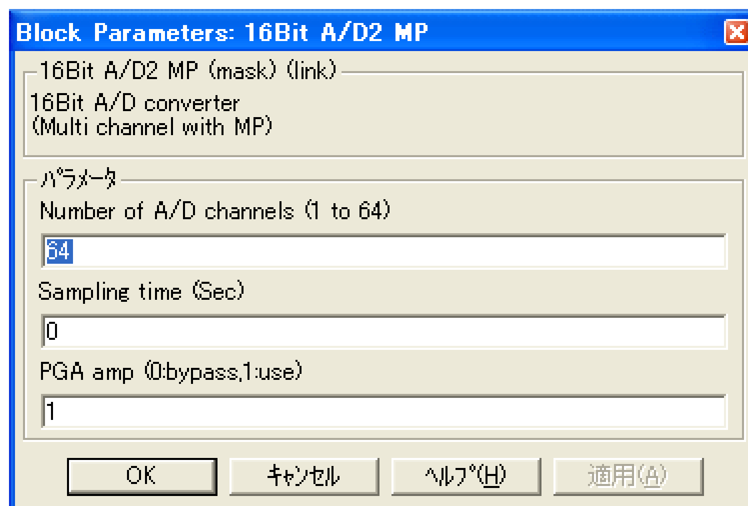
２．１０．７.　　多チャンネル A/D 入力（MP 付き）デバイスドライバブロック

多チャンネル A/D 入力（MP＝マルチプレクサ付き）デバイスドライバブロックは、複数チャンネルの A/D から一括してデータを入力します。入力の対象となる A/D チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。

このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

このデバイスドライバはマルチプレクサ（オプション）が有る場合のみ利用可能です。

図 ２．１０－５　ADSP324-13 A/D input （M） MP



- Number of A/D channels （1 to 64）
  入力するA/Dのチャンネル数を１～６４の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１０－２ （97ページ） を参照してください。

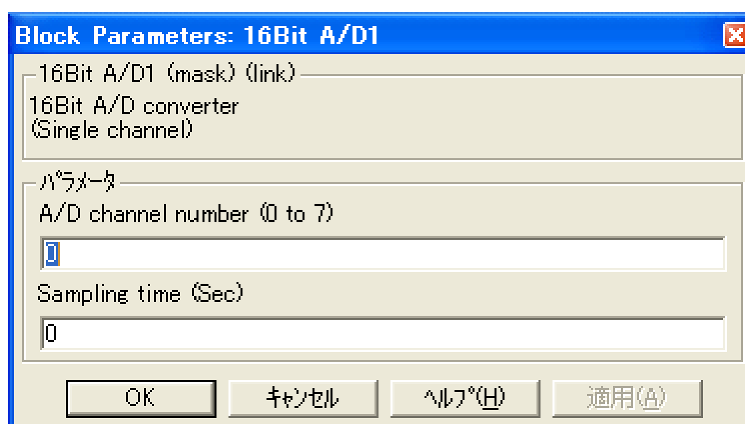
- Sampling time （Sec）
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムで A/D 変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎に A/D 変換が行われます。

- PGA amp （0:bypass，1:use）
  ボード上のPGAを使用しているか否を設定します。通常は１を設定してください。AD変換の高速化の為、PGAのバイパス（２．１０．３参照）を実施した場合のみ０としてください。

２．１０．８．　　１チャンネル D/A 出力デバイスドライバブロック

１チャンネル D/A 出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．１０−６　ADSP324-13 D/A output　(S)



- D/A channel number (0 to 7)
  出力するD/Aのチャンネル番号を０～７の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１０−１　(97ページ) を参照してください。

２．１０．９．　多チャンネル D/A 出力デバイスドライバブロック

多チャンネル D/A 出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。出力の対象となる D/A チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへの入力ベクトルの最初の要素がチャンネル番号０の D/A へ出力されます。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．１０－７　ADSP324-13 D/A output （M）

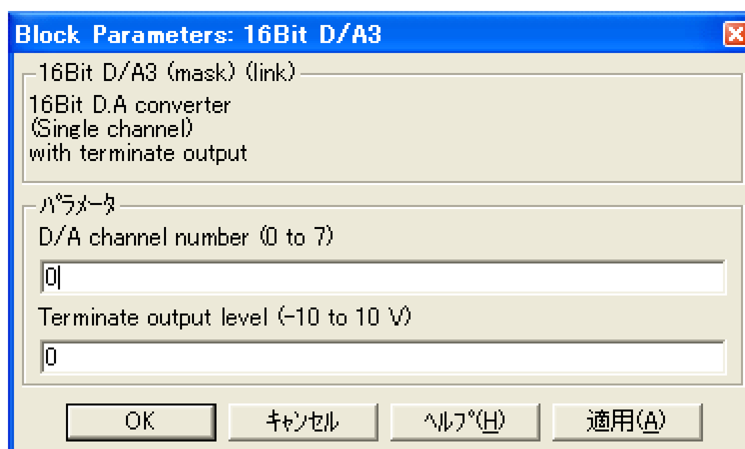


- Number of D/A channels （1 to 8）
  出力するD/Aのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１０－１　(97ページ)　を参照してください。

２．１０．１０．　終了時出力指定　１チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き１チャンネル D/A 出力デバイスドライバブロックは､スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．１０−８　ADSP324-13 D/A output (S) Terminate output



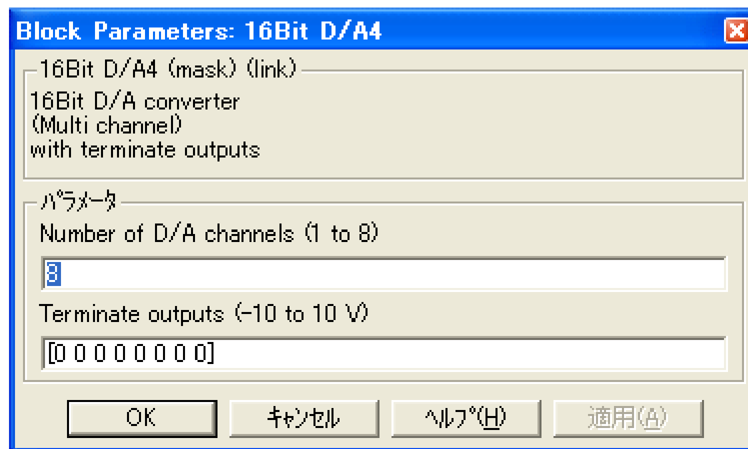
- D/A channel number (0 to 7)
  出力するD/Aのチャンネル番号を０～７の整数で指定します｡チャンネル番号とハードウェアとの対応は、表　２．１０−１ (97ページ) を参照してください。

- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際に D/A から出力する値を設定します。単位は[Volt]です。

２．１０．１１．　終了時出力指定 多チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き 多チャンネル D/A 出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。出力の対象となる D/A チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。
このブロックへの入力ベクトルの最初の要素がチャンネル番号０の D/A へ出力されます。
このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．１０−９　ADSP324-13 D/A output (M) Terminate output



- Number of D/A channels (1 to 8)
  出力するD/Aのチャンネル数を１～８の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１０−１ (97ページ) を参照してください。

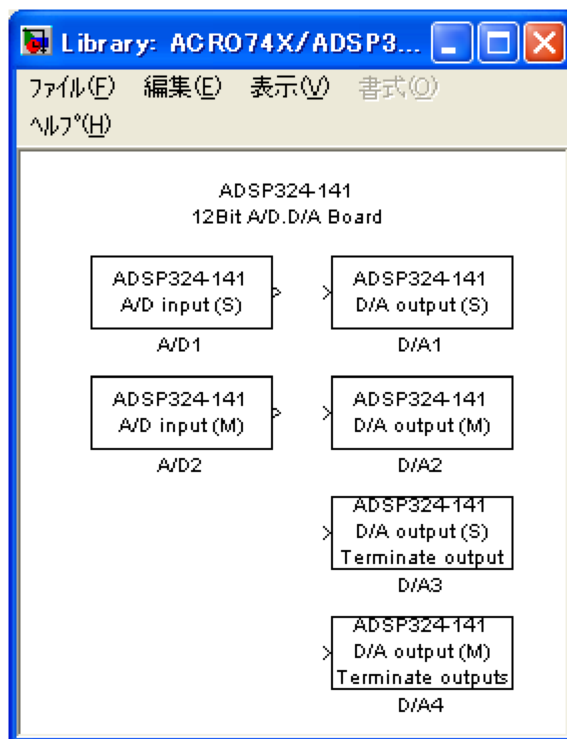
- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際に D/A から出力する値をベクトルで設定します。単位は[Volt]です。

２．１１．　ADSP324-141 デバイスドライバブロック

２．１１．１．　　概要

図　２．１１−１に、ADSP324-141（１２ビットA/D・D/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図　２．１１−１　ADSP324-141 ドライバウィンドウ



この中にはADSP324-141 とのインターフェースを取るための６つのデバイスドライバブロックが登録されています。

- A/D input (S)　　１チャンネルの A/D 入力デバイスドライバブロックです。
- A/D input (M)　　多チャンネルの A/D 入力デバイスドライバブロックです。
- D/A input (S)　　１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M)　　多チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (S) Terminate output
  終了時の出力指定が行える、１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M) Terminate output
  終了時の出力指定が行える、多チャンネルの D/A 出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

終了時出力指定の行えるデバイスドライバは、実時間モデルが停止する際に、予め初期化ダイアログで指定した値を出力します。リアルタイムモニター、UIL、ユーザ定義関数等で強制停止させた場合にも終了時出力は行われます。それ以外のブロックは実時間モデルが停止する直前の値を出力し続けます。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

２．１１．２．　チャンネル番号とボードとの対応

ADSP324-141 は最大４台使用でき、１台のボードにA/D、D/Aが各１６及び８チャンネル（オプション実装時１２チャンネル）実装されています。ADSP324-141 の各デバイスドライバブロックはこれらのチャンネルを連続した論理チャンネル番号として取り扱います。論理チャンネル番号とハードウェアとの対応は、表 ２．１１−１ 及び表 ２．１１−２ の通りです。ボードの１～４台目の区別についてはハードウェアのセットアップ (5 ページ) を参照してください。

表 ２．１１−１　入力チャンネル番号と ADSP324-141 ボードとの対応

ボード	IN0	IN1	…	IN14	IN15
１台目	０	１	…	１４	１５
２台目	１６	１７	…	３０	３１
３台目	３２	３３	…	４６	４７
４台目	４８	４９	…	６２	６３

表 ２．１１−２　出力チャンネル番号と ADSP324-141 ボードとの対応

(D/A 標準実装の場合)

ボード	OUT0	OUT1	…	OUT6	OUT7
１台目	０	１	…	６	７
２台目	８	９	…	１４	１５
３台目	１６	１７	…	２２	２３
４台目	２４	２６	…	３０	３１

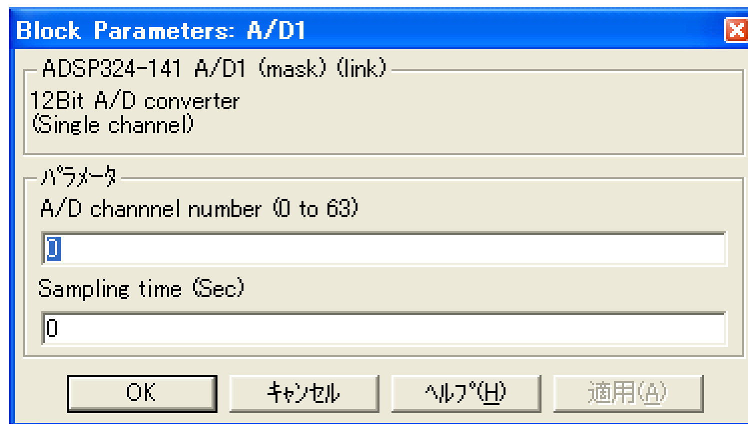
(D/A の CH 増設オプション有りの場合)

ボード	OUT0	OUT1	…	OUT10	OUT11
１台目	０	１	…	１０	１１
２台目	１２	１３	…	２２	２３
３台目	２４	２５	…	３４	３５
４台目	３６	３７	…	４６	４７

２．１１．３．　1チャンネル A/D 入力デバイスドライバブロック

１チャンネル A/D 入力デバイスドライバブロックは、指定されたチャンネル番号の A/D からデータを入力します。このブロックからの出力はスカラーとなります。
このブロックからの出力データの単位は電圧値 [V] です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

図 ２．１１－２　ADSP324-141 A/D input (S)



- A/D channel number (0 to 63)
  入力するA/Dのチャンネル番号を０～６３の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１１－１ (109ページ) を参照してください。

- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

２．１１．４．　　多チャンネル A/D 入力デバイスドライバブロック

多チャンネル A/D 入力デバイスドライバブロックは、複数チャンネルの A/D から一括してデータを入力します。入力の対象となる A/D チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックからの出力は、指定されたチャンネル数の幅を持つベクトルとなります。ベクトルの最初の要素がチャンネル番号０からのデータです。
このブロックからの出力データの単位は電圧値［V］です。センサー等の物理単位への変換はデータ入力後 Gain ブロックを使って行ってください。

図　２．１１－３　ADSP324-141 A/D input (M)



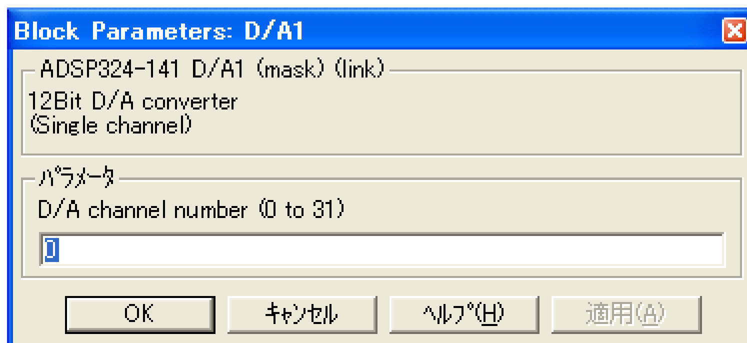
- Number of A/D channels (1 to 64)
  入力するＡ／Ｄのチャンネル数を１～６４の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１１－１（109ページ）を参照してください。

- Sampling time (Sec)
  サンプリングの時間間隔を指定します。単位は秒です。ここで指定したサンプリングタイムと実時間モデルのステップサイズが異なる場合は、ここで指定されたサンプリングタイムでＡ／Ｄ変換が行われます。特殊な場合として、ここに０を入力すると、実時間モデルのステップサイズ毎にＡ／Ｄ変換が行われます。

２．１１．５.　　１チャンネル D/A 出力デバイスドライバブロック

１チャンネル D/A 出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．１１－４　ADSP324-141 D/A output (S)

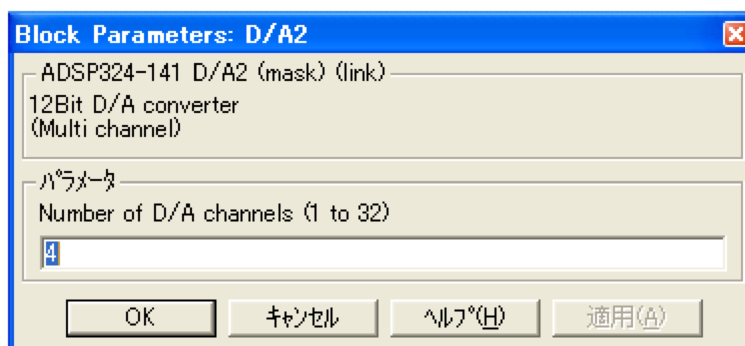


- D/A channel number (0 to 31)
  出力するD/Aのチャンネル番号を０～３１の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１１－２ (109ページ) を参照してください。

図　２．１１－５多チャンネル D/A 出力デバイスドライバブロック

多チャンネル D/A 出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。出力の対象となる D/A チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０の D/A へ出力されます。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．１１－６　ADSP324-141 D/A output (M)

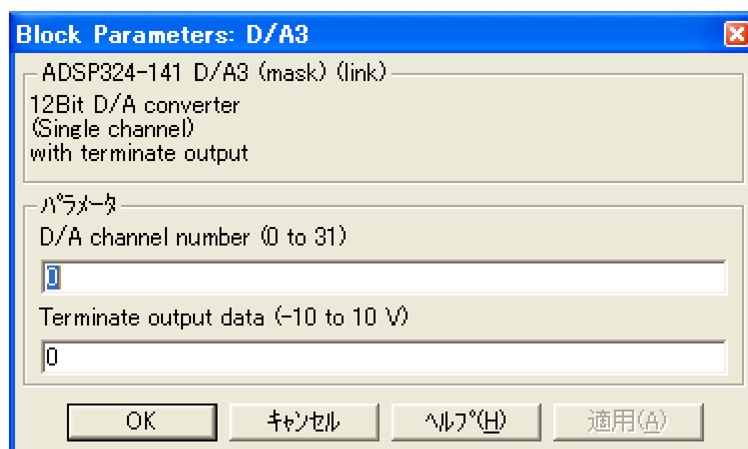


- Number of A/D channels (1 to 32)
  出力するD/Aのチャンネル数を１～３２の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１１－２ (109ページ) を参照してください。

２．１１．６．　終了時出力指定１チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き１チャンネル D/A 出力デバイスドライバブロックは､スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。このブロックは実時間モデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[V]です｡物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．１１-７　ADSP324-141 output (S) Terminate output



- D/A channel number (0 to 31)
  出力するD/Aのチャンネル番号を０～３１の整数で指定します｡チャンネル番号とハードウェアとの対応は、表 ２．１１-２ (109ページ) を参照してください。

- Terminate output data (-10 to 10V)
  リアルタイムモデルが終了する際に D/A から出力する値を設定します。単位は[Volt]です。

２．１１．７．　終了時出力指定 多チャンネル D/A 出力デバイスドライバブロック

終了時出力指定付き多チャンネル D/A 出力デバイスドライバブロックは､ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０の D/A へ出力されます。このブロックはモデルが停止する際に、予め指定した電圧値を出力します。
このブロックへの入力データの単位は電圧値[V]です｡物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．１１−８　ADSP324-141 D/A output （M） Terminate output

Block Parameters: D/A4
ADSP324-141 D/A4 (mask) (link)
12Bit D/A converter
(Multi channel)
with terminate outputs
ﾊﾟﾗﾒｰﾀ
Number of D/A channels (1 to 32)
32
Terminate output datas (-10 to 10 V)
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
OK　ｷｬﾝｾﾙ　ﾍﾙﾌﾟ(H)　適用(A)

- Number of D/A channels （1 to 32）
  出力するD/Aのチャンネル数を１～３２の整数で指定します｡先頭チャンネル番号は０で固定です｡
  チャンネル番号とハードウェアとの対応は、表 ２．１１−２ （109ページ） を参照してください。

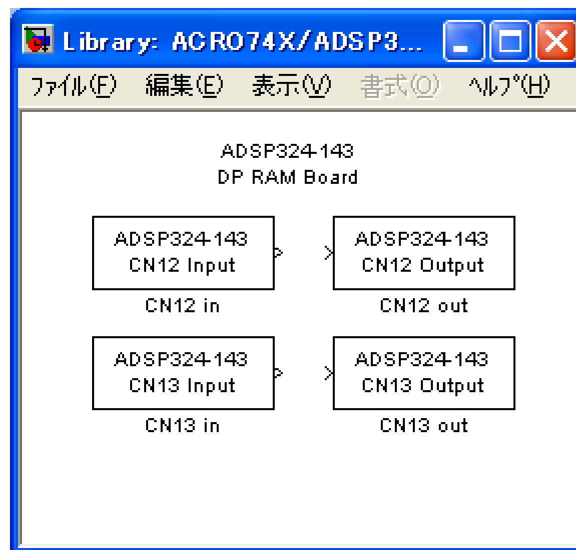
- Terminate output data （-10 to 10V）
  リアルタイムモデルが終了する際に D/A から出力する値をベクトルで設定します。単位は[Volt]です。

２．１２．　ADSP324-143 デバイスドライバブロック

２．１２．１．　　概要

図　２．１２－１に、ADSP324-143（DP　RAMボード）用デバイスドライバウィンドウを開いた所を示します。

図　２．１２－１　ADSP324-143 ドライバウィンドウ



この中には ADSP324-143 とのインターフェースを取るための４つのデバイスドライバブロックが登録されています。

- CN12 input　　前段 ADSP674 との間にある DPRAM を使い、前段 ADSP674 からデータを受け取るデバイスドライバブロックです。
- CN13 input　　後段 ADSP674 との間にある DPRAM を使い、後段 ADSP674 からデータを受け取るデバイスドライバブロックです。
- CN12 output　　前段 ADSP674 との間にある DPRAM を使い、前段 ADSP674 へデータを渡すデバイスドライバブロックです。
- CN13 output　　後段 ADSP674 との間にある DPRAM を使い、後段 ADSP674 へデータを渡すデバイスドライバブロックです。

＊　前段・後段ADSP674 については、**“ADSP674 の前段と後段の意味”**（123ページ）を参照してください。

２．１２．２．　　コネクタとボードとのメモリマップ

ADSP324-143 は、ADSP674１枚に対して最大４台使用できます。コネクタ番号とADSP674 のメモリ対応は、表 ２．１２−１の通りです。ボードの１～４台目の区別については 表 １．１−１　ボードアドレス設定表 (5 ページ) を参照してください。

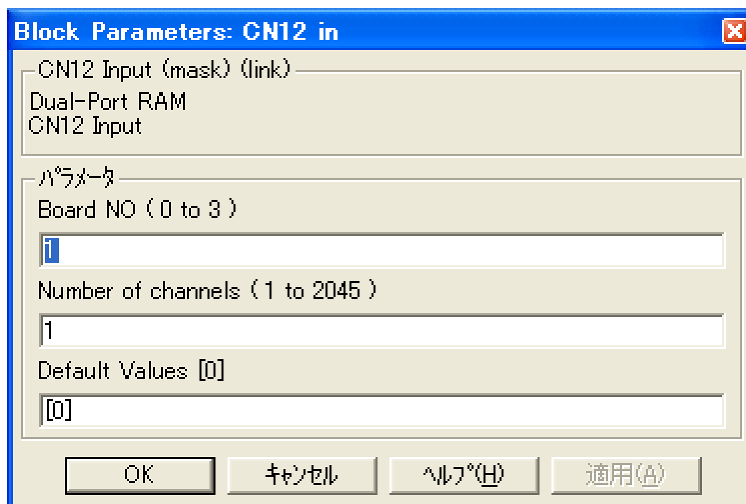
表 ２．１２−１　コネクタと ADSP324-143 ボードメモリマップ

ボード	ボード番号	ＣＮ１２	ＣＮ１３
１台目	0	908000ｈ～908FFFh	909000ｈ～909FFFｈ
２台目	1	90C000h～90CFFFｈ	90D000h～90DFFFｈ
３台目	2	910000ｈ～910FFFｈ	911000ｈ～911FFFｈ
４台目	3	914000ｈ～914FFFｈ	915000ｈ～915FFFｈ

２．１２．３．　CN12　Input デバイスドライバブロック

CN12　Input デバイスドライバは、コネクタ１２を使用し、ADSP674 と ADSP674 を接続するパイプラインボード（オプション）を用いて、前段 ADSP674 との間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ADSP674 間の通信をホストコンピュータのバスを介さずに直接 ADSP674 間で交信することができます。

図　２．１２－２　ADSP324-143 CN12　Input



- Board　NO (0 to 3)
  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表　２．１２－１(117ページ)を参照してください。

- Number　of　channels (1 to 2045)
  相手 ADSP674 から受け取るデータのチャネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

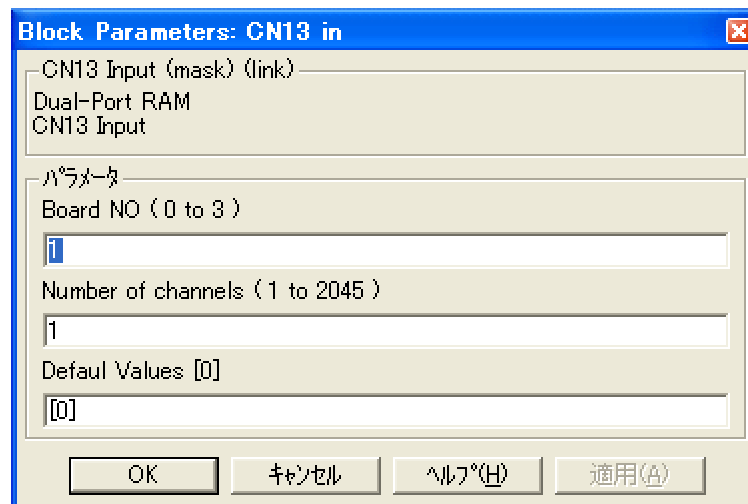
- Default Values [0]
  相手 ADSP674 が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル（[ ]）を与えます。空ベクトル時は、第二引数の幅のデータ０ベクトルを与えたことと同一になります。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“ADSP674 間DP通信アービトレーション”（122ページ）を参照してください。
又、前段・後段ADSP674 については、“ADSP674 の前段と後段の意味”（123ページ）を参照してください。

２．１２．４．　CN13　Input デバイスドライバブロック

CN13　Input デバイスドライバは、コネクタ１３を使用し、ADSP674 と ADSP674 を接続するパイププラインボード（オプション）を用いて、後段 ADSP674 との間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ADSP674 間の通信をホストコンピュータのバスを介さずに直接 ADSP674 間で交信することができます。

図　２．１２－３ ADSP324-143 CN13　Input



- Board　NO　(0 to 3)
  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表　２．１２－１(117ページ)を参照してください。

- Number　of　channels　(1 to 2045)
  相手 ADSP674 から受け取るデータのチャネル数を設定します。データはここで設定した幅のベクトルとしてこのブロックから出力されます。

- Default Values　[0]
  相手 ADSP674 が未だ一度もデータを送信していない段階でこのブロックから出力される値を設定します。第二引数で指定した点数の幅をもつ定数ベクトル又は空ベクトル（[ ]）を与えます。空ベクトル時は、第二引数の幅のデータ０ベクトルを与えたことと同一になります。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“**ADSP674 間DP通信アービトレーション**”（122ページ）を参照してください。
又、前段・後段ADSP674 については、“**ADSP674 の前段と後段の意味**”（123ページ）を参照してください。

２．１２．５．　CN12　Output デバイスドライバブロック

CN12　Output デバイスドライバは、コネクタ１２を使用し、ADSP674 と ADSP674 を接続するパイプラインボード（オプション）を用いて、前段 ADSP674 との間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ADSP674 間の通信をホストコンピュータのバスを介さずに直接 ADSP674 間で交信することができます。

図　２．１２－４ ADSP324-143 CN12　Output



- Board　NO　(0 to 3)
  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表　２．１２－１ (117ページ) を参照してください。

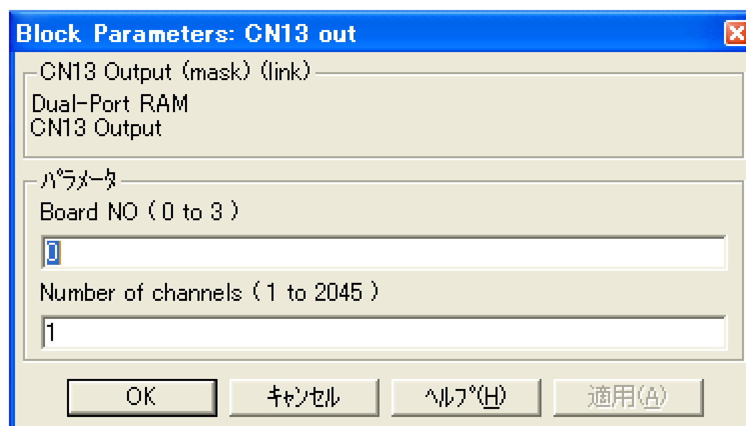
- Number　of　channels　(1 to 2045)
  相手 ADSP674 へ渡すデータのチャネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、**“ADSP674 間DP通信アービトレーション”**（122ページ）を参照してください。
又、前段・後段ADSP674 については、**“ADSP674 の前段と後段の意味”**（123ページ）を参照してください。

２．１２．６．　CN13　Output デバイスドライバブロック

CN13　Output デバイスドライバは、コネクタ１３を使用し、ADSP674 と ADSP674 を接続するパイプラインボード（オプション）を用いて、後段 ADSP674 との間で非同期にデータ交換を行うためのブロックです。マルチプロセッシングを構成した場合、ADSP674 間の通信をホストコンピュータのバスを介さずに直接 ADSP674 間で交信することができます。

図 ２．１２−５ ADSP324-143 CN13　Output



- Board　NO (0 to 3)
  ADSP324-143 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表 ２．１２−１(117ページ)を参照してください。

- Number　of　channels (1 to 2045)
  相手 ADSP674 へ渡すデータのチャネル数を設定します。データはここで設定した幅のベクトルをこのブロックに入力します。

複数のデータを渡す場合の、データの同時性は保証されています。詳しくは、“ADSP674 間DP通信アービトレーション”（122ページ）を参照してください。
又、前段・後段ADSP674 については、“ADSP674 の前段と後段の意味”（123ページ）を参照してください。

２．１２．７．　　ADSP674 間 DP 通信アービトレーション

ADSP674 間の DPRAM 通信デバイスブロックによる交信は、データの同時性（ブロックから取り出したデータ（送ったデータ）が全て同一時刻である事）を保証しています。つまり、「データの前半が後半より１時刻（ステップサイズ）分、昔のデータであった」等ということが無いように、送る側と受ける側でフラグを用いたソフトウェア同期処理を行っています。
これは、複数の ADSP674 が異なる周期で計算を行う、非同期処理間でデータを交換する為には必要な機構です。

本ソフトウェアでは、データの同時性は、以下の考え方によりアービトレーションしています。
４、書き込み側の要求が先に発生した場合
読み出し側は書き込み側が完了するまで待機して後読み出す。
５、読み出し側の要求が先に発生した場合
書き込み側は読み出しみ側が完了するまで待機して後書き込む。
６、両方の要求が同時に発生した場合
読み出し側は書き込み側が完了するまで待機して後読み出す。

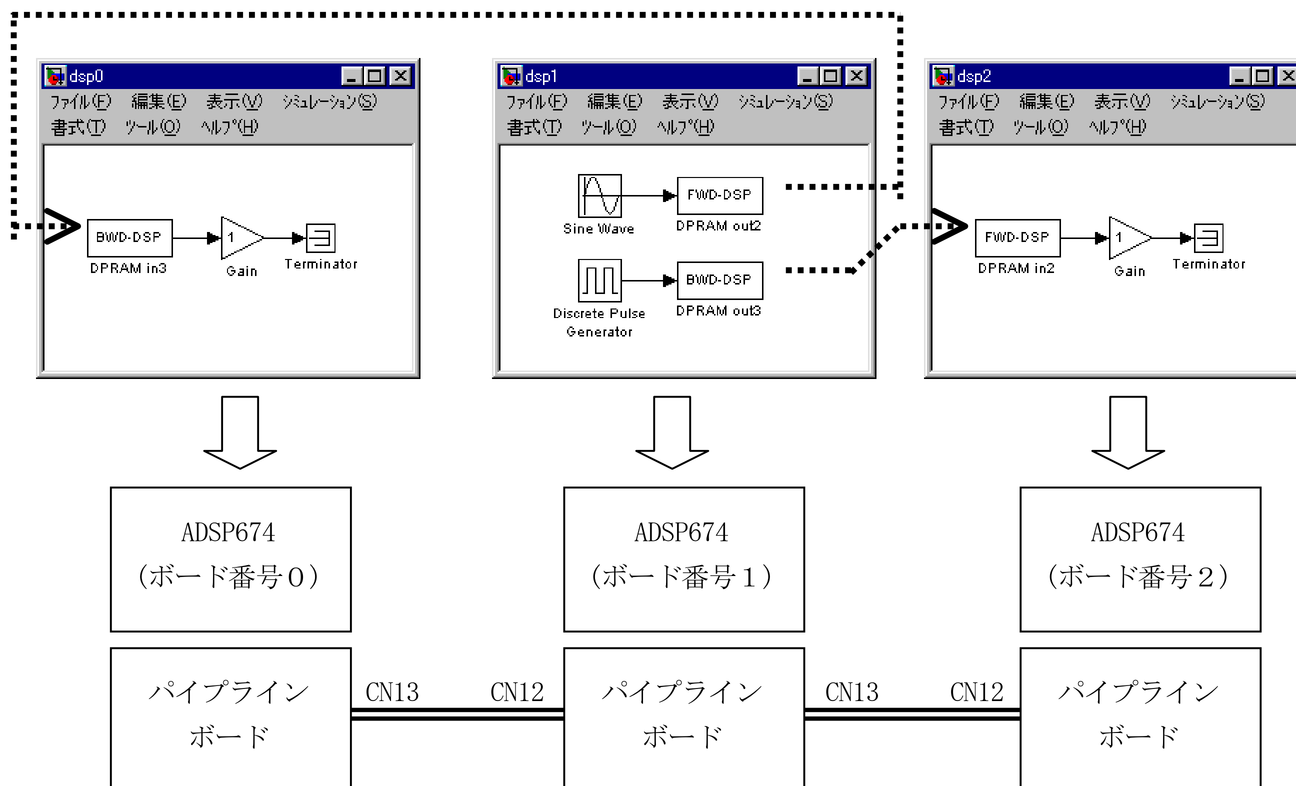
上記３項は、同時の場合は、最新のデータを渡す為、読み出し側が待機します。

２．１２．８．　ADSP674 の前段と後段の意味

ADSP674 間の DPRAM 通信デバイスブロックには、前段（FWD）と後段（BWD）があります。前段とは当該 ADSP674 に接続されているパイプラインボード（ADSP324-143 等）のコネクタ CN12 の先に接続された ADSP674 であり、後段とは同 CN13 の先に接続されている ADSP674 の事を表します。つまり、前段・後段の関係は、パイプラインボードの接続がどうなっているかにより決定され、個々の ADSP674 のボード番号には全く関係ありません。

以下に簡単な例を示します。説明の便宜上各 ADSP674 にはボード番号０、１、２と番号を付けたとします。下の図では、真中の ADSP674（ボード番号１）から見て前段とは、同ボードに接続されたパイプラインボードの CN12 に接続された ADSP674 ですから、ボード番号０が前段になり、同様に後段は、同ボードに接続されたパイプラインボードの CN13 に接続された ADSP674 ですから、ボード番号２が後段になります。

図のブロック線図を結ぶ破線矢印 ···> は信号の流れを表しています。

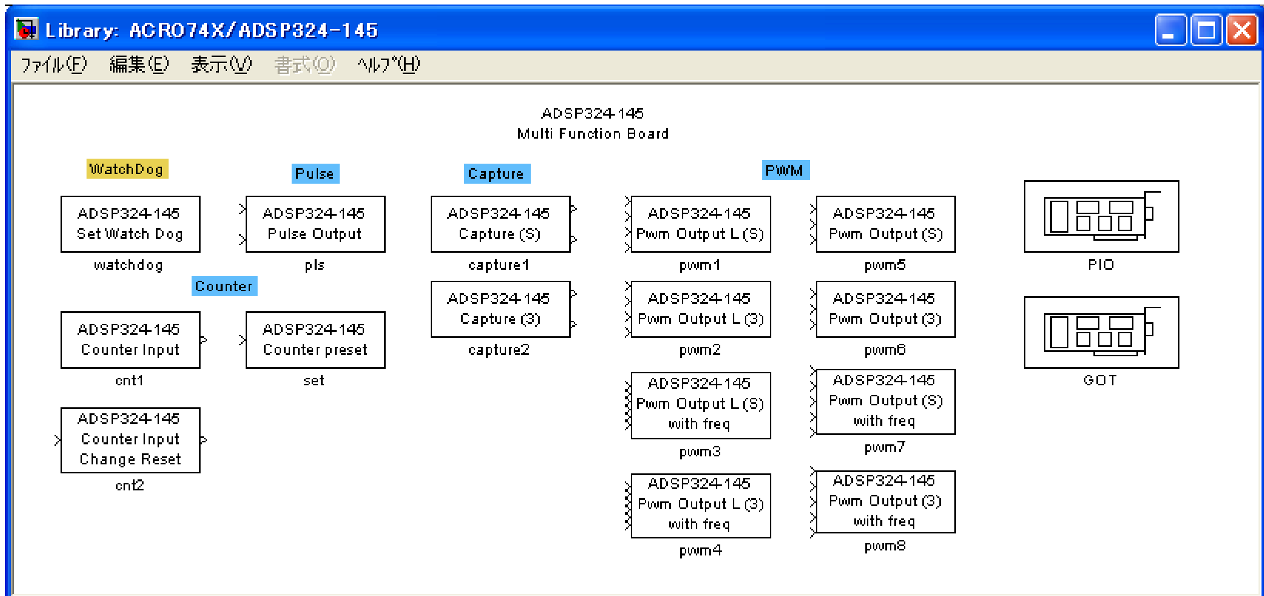


２．１３．ADSP324-145 デバイスドライバブロック

２．１３．１．　概要

図　２．１３－１に、ADSP324-145 用デバイスドライバウィンドウを開いた所を示します。

図　２．１３－１　ADSP324-145 ドライバウィンドウ



この中には ADSP324-145 とのインターフェースを取るための２２のデバイスドライバブロックが登録されています。

- watch dog　　ウォッチドグタイマ
- Pulse Output　　パルス出力デバイスドライバです。
- Counter Input　　プリセット付カウンタデバイスドライバです。
- Counter preset　　カウンタープリセットドライバです。
- Capture[S]　　単相モードのキャプチャデバイスドライバです。
- Capture[3]　　３相モードのキャプチャデバイスドライバです。
- Pwm Output[S]　　単相モードのＰＷＭ出力デバイスドライバです。
- Pwm Output[3]　　３相モードのＰＷＭ出力デバイスドライバです。
- PIO Set Time　　PIO のﾊﾝﾄﾞﾘﾝｸﾞ信号の ON/遅延時間設定デバイスドライバです。
- PIO bit In[x]　　ビット入力デバイスドライバです。
- PIO Paralell In[x]　　複数ビット同時入力デバイスドライバです。
- PIO bit Out[x]　　ビット出力デバイスドライバです。
- PIO Paralell Out[x]　　複数ビット同時出力デバイスドライバです。
- GOT　　GOT 操作用デバイスドライバです。

２．１３．２．　コネクタとボードとのメモリマップ

ADSP324-145 は、ADSP674１枚に対して最大４台使用できます。コネクタ番号とADSP674 のメモリ対応は、表 ２．１３−１ の通りです。ボードの１〜４台目の区別についてはハードウェアのセットアップ（**エラー! ブックマークが定義されていません。**ページ）を参照してください。

表 ２．１３−１　コネクタと ADSP324-145 ボードメモリマップ

ボード	ボード番号	ADSP324-145		
		ADSP324-00A	ADSP674 ｼﾘｰｽﾞ	ADSP674 ｼﾘｰｽﾞ
１台目	0	00901000h	03004000h	82404000h
２台目	1	00901200h	03004800h	82404800h
３台目	2	00901400h	03005000h	82405000h
４台目	3	00901600h	03005800h	82405800h

表 ２．１３−２　ADSP324-145 カウンタチャネル番号対応

	カウンタチャンネル番号				
ボード	ｶｳﾝﾀｰ０	ｶｳﾝﾀｰ１	ｶｳﾝﾀｰ２	ｶｳﾝﾀｰ３	絶縁ポート番号
１台目	０	１	２	３	０
２台目	４	５	６	７	１
３台目	８	９	１０	１１	２
４台目	１２	１３	１４	１５	３

表 ２．１３−３　ADSP324-145 キャプチャ・ＰＷＭ・パルスチャネル番号対応

	キャプチャ・ＰＷＭ・パルスチャンネル番号			
ボード	ＣＨ０	ＣＨ１	ＣＨ２	ＣＨ３
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

表　２．１３－４　　ADSP324-145PIO ポート番号対応

ボード	ポート０ OUT	ポート０ IN	ポート２ OUT	ポート２ IN
１台目	０	１	２	３
２台目	４	５	６	７
３台目	８	９	１０	１１
４台目	１２	１３	１４	１５

表　２．１３－５　　ADSP324-145 ボード PIO とビット番号の対応

入出力信号名	ビット番号 注１	パラレル 注２	ポート単位 注３
SIG031 / SIG331	31	最後	2^{31}
SIG030 / SIG330	30	最後の１つ前	2^{30}
・	・	・	・
・	・	・	・
SIG006 / SIG306	6	・	2^{6}
SIG005 / SIG305	5	・	2^{5}
SIG004 / SIG304	4	・	2^{4}
SIG003 / SIG303	3	・	2^{3}
SIG002 / SIG302	2	・	2^{2}
SIG001 / SIG301	1	先頭の次	2^{1}
SIG000 / SIG300	0	先頭	2^{0}

注１：ビット単位の入出力の場合、各ブロックの初期化ダイアログで指定するビット番号に相当します。

注２：パラレル入出力の場合、ブロック線図との間で受け渡すベクトルの各要素と入出力信号との関係は表の通りとなります。終了処理付きの場合の終了値も同様です。

注３：ポート単位の入出力の場合、各ブロックとブロック線図との間で受け渡すデータは、各ビットをここに示した重みで加算された値となります。

表　２．１３－６　PIO ハンドリング方式

	ハンドリング方式
０	ハンドリングなし
1	ホールド方式
2	Strobe / Ack 方式
3	Write Strobe 方式

★ハンドリング方式、ＰＩＯ信号論理の詳細は、“ADSP324-145 ハードウェア・ユーザーズ・マニュアル” 機能説明（パラレル入出力　ハンドリング方式）等を参照してください。

２．１３．３．　WatchDog デバイスドライバブロック

WatchDog デバイスドライバは、ウォッチドグタイマを使用可能にします。

図　２．１３－２　ADSP324-145 WatchDog



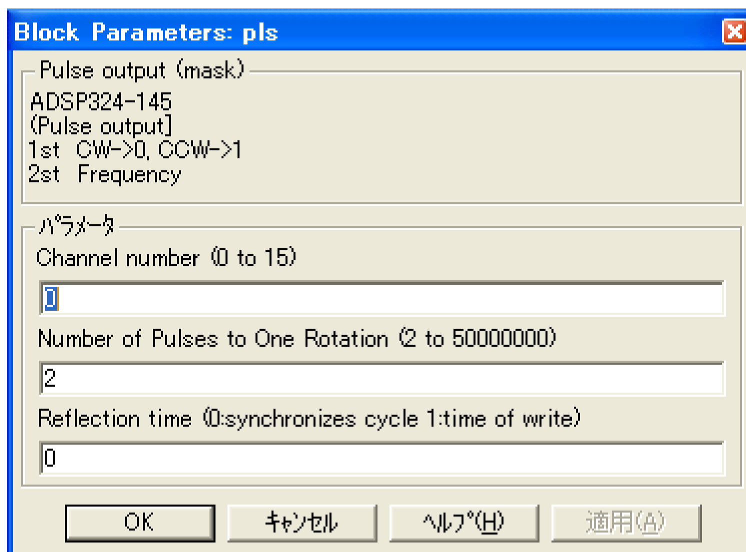
- Board　NO (0 to 3)
  ADSP324-145 のボード番号を０～３の整数で指定します。ボード番号とハードウェアとの対応は、表　２．１３－１(125ページ)を参照してください。

- Surveillance　time〔0 to 1000000mSec〕
  ウォッチドグタイマの監視周期時間を 0～1000000 mSec で設定してください。

２．１３．４．　Pulse output デバイスドライバブロック

Pulse output デバイスドライバは、第一入力として、回転方向（０：正転<CW>、１逆転<CCW>)、第二入力としてパルス周波数（０．０１Hz～６MHz）を入力すると、指定チャネルへパルス出力します。

図 ２．１３－３　ADSP324-145 Pulse output



- Channel number (0 to 15)
  ADSP324-145 のパルスチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３－３を参照してください。

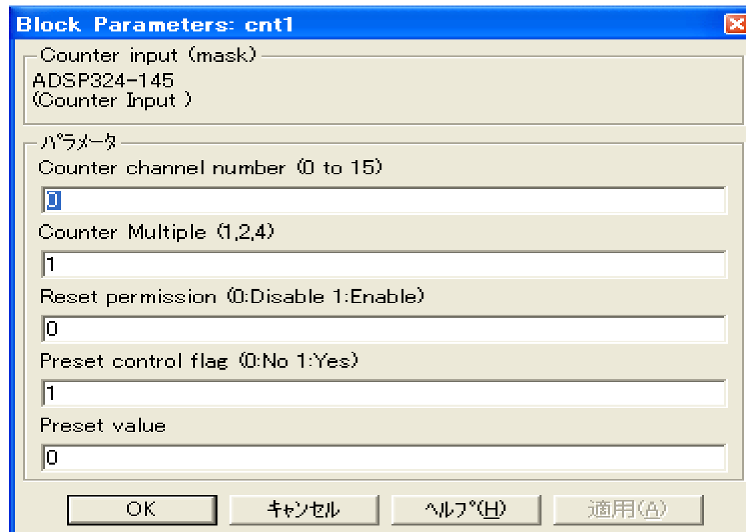
- Number of Pulses to One Rotation 〔1 to 50000000〕
  １回転のパルス数を指定します。パルス数とは、Z 相の出力を A・B 相パルスの何パルスに１回出力するかを設定します。

- Reflection timing〔0: synchronizes cycle 1: time o fwrite〕
  パルス設定値の反映タイミングを０：周波数に同期、１：書込み時のいずれかを指定します。

2．1 3．5．　Counter lnput デバイスドライバブロック

Counter lnput デバイスドライバブロックは、任意のチャンネル番号のカウンタの値を読み込みます。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンタは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行った場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。又リセットを有効にした場合リセット条件が満たされた時カウンタがリセットされます。

図 2．1 3－4　ADSP324-145 Counter lnput



- Counter channel number (0 to 15)
  入力の対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表 2．1 3－2 を参照してください。

- Counter Multiple (1, 2, 4)
  カウンタの逓倍を　１：×１、２：×２、４：×４　のいずれかより選択してください。

- Reset permission (0:Disable 1: Enable)
  カウンタリセット信号の許可を設定してください。（０：未許可、１：許可）
  詳細は、ADSP324-145 ハードウェアマニュアルを参照してください。

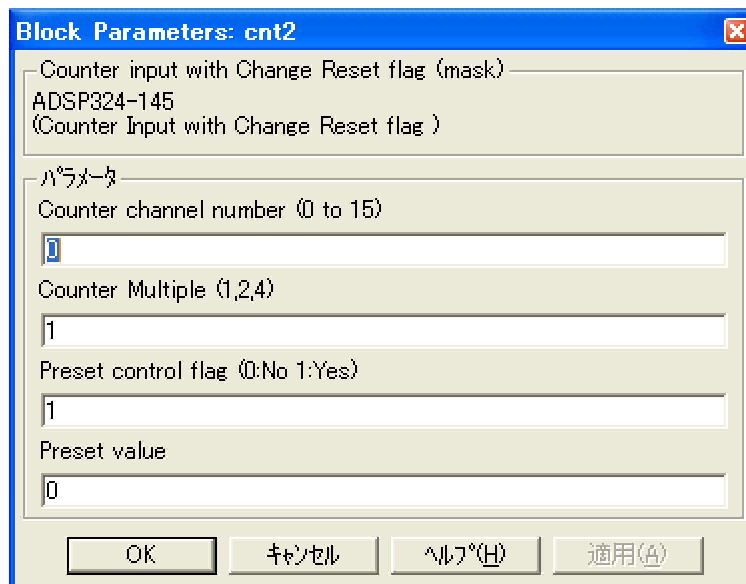
- Preset controll flag (0:No 1:Yes)
  初期化時にカウンタ値のプリセットを行うか否かを決定します。０又は１を設定してください。０を設定した場合、プリセットは行いません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。

- Preset value
  Preset controll flag に１を設定した場合、ここに設定した値がカウンタの初期値として採用されます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

２．１３．６．　Counter Input with Change reset flag デバイスドライバブロック

Counter Input with Change Reset flag デバイスドライバブロックは、任意のチャンネル番号のカウンタの値を読み込みます。リセット許可（０：未許可　１：許可）をスカラーで入力します。このブロックの出力はスカラーです。ADSP324-145 ボードの指定されたカウンタは符号付き３２ビット整数として読み込まれ、スカラーに変換されて出力されます。プリセットの指定を行った場合、初期化の際に指定した値にプリセットされます。プリセットは初期化の際に１回だけ行なわれます。

図　２．１３－５　ADSP324-145 Counter Input with Change Reset flag



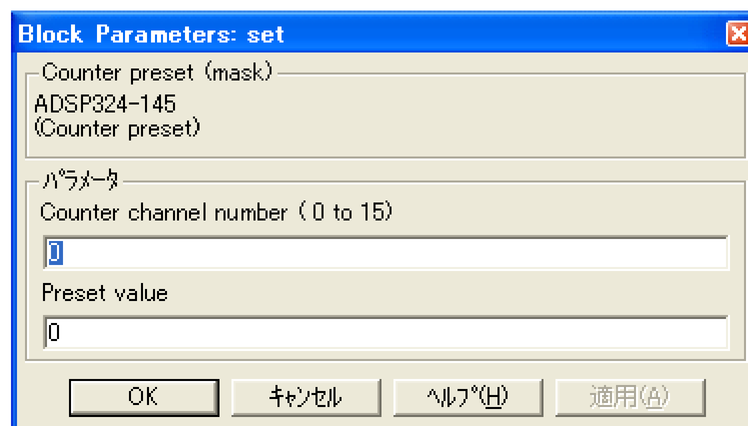
- Counter channel number（0 to 15）
  入力の対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表　２．１３－２を参照してください。
- Counter Multiple（1,2,4）
  カウンタの逓倍を　１：×１、２：×２、４：×４　のいずれかより選択してください。
- Preset controll flag（0:No 1:Yes）
  初期化時にカウンタ値のプリセットを行うか否かを決定します。０又は１を設定してください。０を設定した場合、プリセットは行いません。１を設定した場合は Preset value に設定した値がカウンタに初期化の際（シミュレーションの開始時）に１度だけ設定されます。
- Preset value
  Preset controll flag に１を設定した場合、ここに設定した値がカウンタの初期値として採用されます。設定できる値は、符号付き３２ビット整数（ストレートバイナリ）で表わせる範囲（-2^{31} ～ $2^{31}-1$）です。

２．１３．７．　Counter preset デバイスドライバブロック

Counter preset デバイスドライバブロックは、任意のチャンネル番号のカウンタを任意の時期にプリセットするデバイスドライバです。プリセット付きカウンタデバイスドライバブロックでは、初期化の際に１度しかプリセットできませんが、このブロックを使用すれば任意の時期に値をプリセットできます。

このブロックには１つのスカラー入力があり、ここに０又は１の値を入力することによりプリセットタイミングを制御します。０が入力されている期間はカウンタはカウント動作を行います。１が入力されるとカウント動作は停止し Preset value で予め設定した値がカウンタにセットされます。

図 ２．１３-６　ADSP324-145 Counter preset



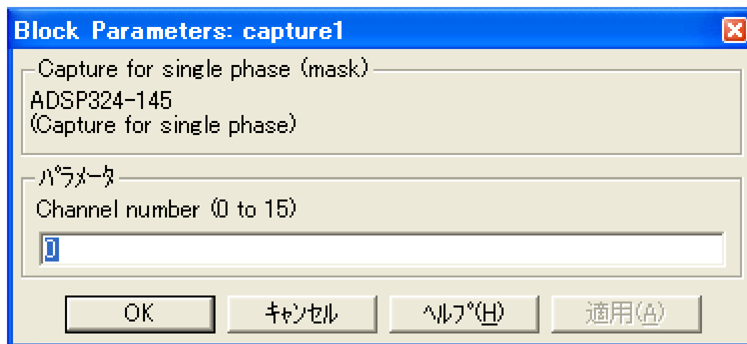
- Counter channel number (0 to 15)
  対象となるカウンタチャネル番号を指定します。チャネル番号とハードウェアとの対応は、表 ２．１３-２を参照してください。

- Preset value
  ここに設定した値がカウンタにセットされます。設定できる値は、符号付３２ビット整数（ストレートバイナリ）で表せる範囲（-2^{31}～$2^{31}-1$）です。

２．１３．８．　Capture(S) デバイスドライバブロック

Capture(S)デバイスドライバは、指定チャネルの単相信号を計測し、第一出力に周期（周波数)、第二出力に UV ON 時間（Duty 比 100.0～-100.0）を出力します。

図 ２．１３－７　ADSP324-145 Capture〔S〕

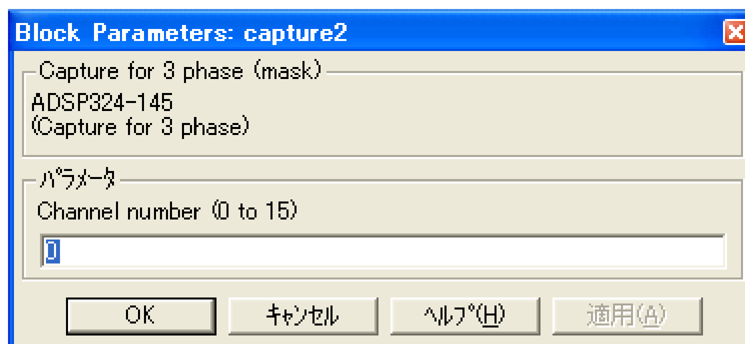


- Channel number (0 to 15)
  ADSP324-145 のキャプチャチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３－３を参照してください。

２．１３．９．　Capture(3)デバイスドライバブロック

Capture(3) デバイスドライバは、指定チャネルの単相信号を計測し、第一出力に周期（周波数)、第二出力に ON 時間（Duty 比 100.0～-100.0）を UV 相・VW 相・WU 相の順にベクトル出力します。

図 ２．１３－８　ADSP324-145 Capture(3)

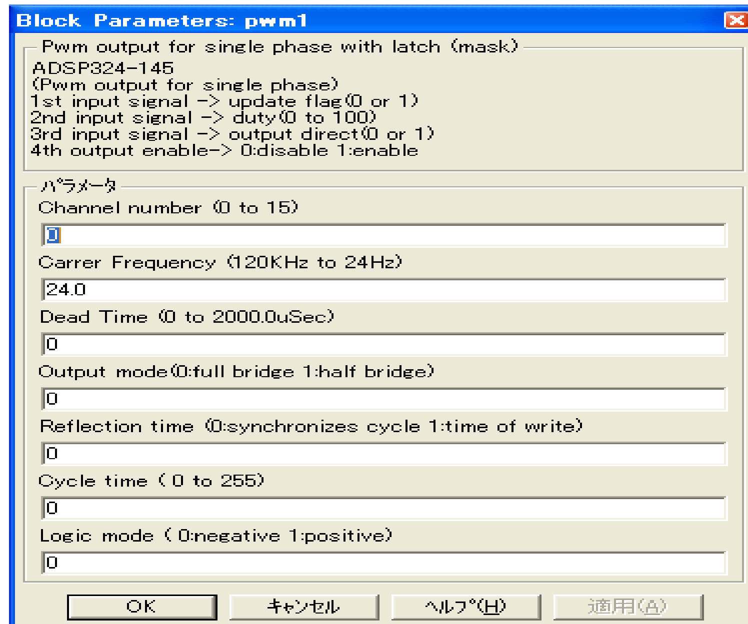


- Channel number (0 to 15)
  ADSP324-145 のキャプチャチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３－３を参照してください。

２．１３．１０．Pwm Output L〔S〕デバイスドライバブロック

Pwm Output〔S〕デバイスドライバは、第一入力として Duty 比の変更有無（１：変更する、０：変更しない）第二入力として Duty 比（0.0～100.0%)、第三入力に出力方向（０：P 側出力、１：N 側出力）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第四入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 ２．１３－９　ADSP324-145 Pwm Output L(S)

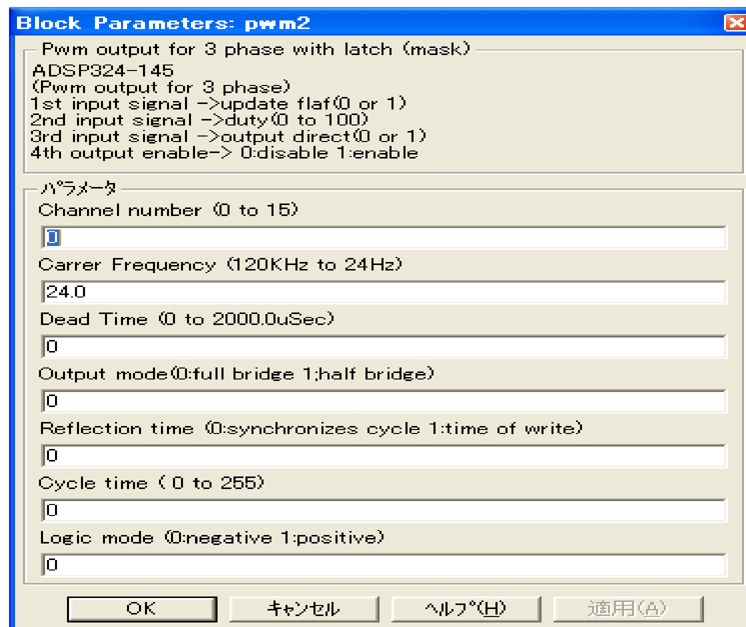


- Channel number (0 to 15)
  ADSP324-145 のPWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３－３を参照してください。
- Carrer Frequency 〔120kHz to 24Hz〕
  キャリア周波数を１２０KHz から２４Hz にて指定してください。
- Dead Time〔0 to 2000.0 μSec〕
  デットタイムを０～２０００．０μSec にて指定してください。
- Output Mode〔0:full bridge　1:half bridge〕
  波形の出力モードを指定してください。
  ０：フルブリッジ出力 １：ハーフブリッジ出力
- Reflection time〔0:synchronizes cycle 1:time of write〕
  設定値の反映タイミングを指定してください。
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）
  １：書きこみ時（モデルの制御周期時に設定値を反映）
- Logic mode〔0:negative 1:positive〕
  同期パルス出力の信号論理を指定してください。
  ０：負論理
  １：正論理

２．１３．１１．Pwm Output L (3)デバイスドライバブロック

Pwm Output(3)　デバイスドライバは、第一入力としてDuty比の変更有無（１：変更する、０：変更しない）第二入力としてDuty比（0.0～100.0%）をU相・V相・W相の順にベクトル入力、第三入力に出力方向（０：P側出力、１：N側出力）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第四入力の出力許可により１を指定時は出力します。[Lなしブロックは第一入力がありません]

図　２．１３－１０　ADSP324-145 Pwm Output L(3)

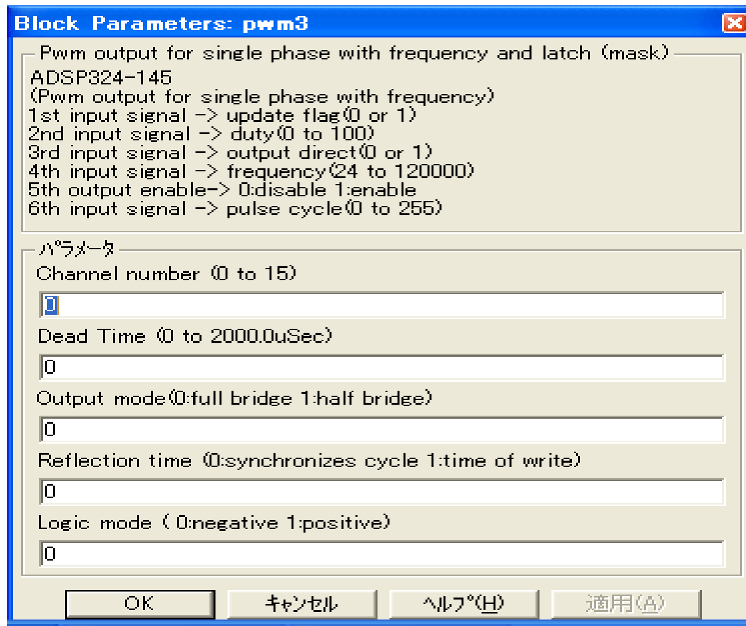


- Channel number (0 to 15)
  ADSP324-145のPWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表　２．１３－３を参照してください。
- Carrer Frequency　[120kHz to 24Hz]
  キャリア周波数を１２０KHzから２４Hzにて指定してください。
- Dead Time [0 to 2000.0　μSec]
  デットタイムを０～２０００．０μSecにて指定してください。
- Output Mode [0:full bridge　1:half bridge]
  波形の出力モードを指定してください。
  ０：フルブリッジ出力　１：ハーフブリッジ出力
- Reflection time [0:synchronizes cycle 1:time of write]
  設定値の反映タイミングを指定してください。
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）
  １：書きこみ時（モデルの制御周期時に設定値を反映）
- Logic mode [0:negative 1:positive]
  同期パルス出力の信号論理を指定してください。
  ０：負論理
  １：正論理

２．１３．１２．Pwm Output L（S）with Freq デバイスドライバブロック

Pwm Output（S）with Freq デバイスドライバは、第一入力として Duty 比の変更有無（１：変更する、０：変更しない）第二入力として Duty 比（0.0～100.0%）、第三入力に出力方向（０：P 側出力、１：N 側出力）、第四入力としてキャリア周波数（120KHz～24Hz）を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにて単相パルスを作成し、第五入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]又、第六入力に同期パルス出力を指定します。

図 ２．１３-１１　ADSP324-145 Pwm Output L（S）with Freq

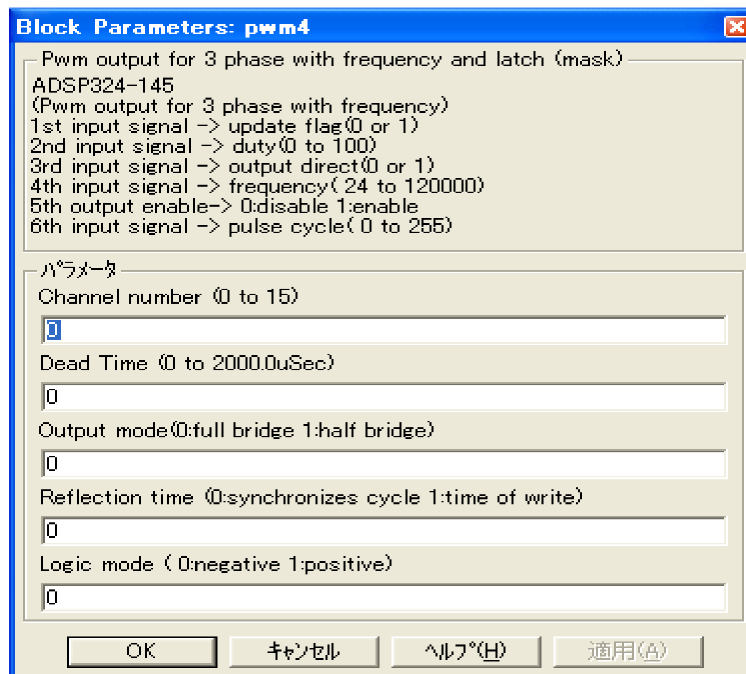


- Channel number（0 to 15）
  ADSP324-145 のPWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３-３を参照してください。
- Dead Time〔0 to 2000.0 μSec〕
  デットタイムを０～２０００．０μSec にて指定してください。
- Output Mode〔0:full bridge　1:half bridge〕
  波形の出力モードを指定してください。
  ０：フルブリッジ出力 １：ハーフブリッジ出力
- Reflection time〔0:synchronizes cycle 1:time of write〕
  設定値の反映タイミングを指定してください。
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映）
  １：書きこみ時（モデルの制御周期時に設定値を反映）
- Logic mode〔0:negative 1:positive〕
  同期パルス出力の信号論理を指定してください。
  ０：負論理
  １：正論理

２．１３．１３．Pwm Output L (3) with Freq デバイスドライバブロック

Pwm Output (3) with Freq デバイスドライバは、第一入力として Duty 比の変更有無（１：変更する、０：変更しない)、第二入力として Duty 比 (0.0～100.0%) をＵ相・Ｖ相・Ｗ相の順にベクトル入力、第三入力に出力方向（０：P 側出力、１：N 側出力)、第四入力としてキャリア周波数 (120KHz～24Hz) を入力すると、指定したチャネルへ指定したキャリア周波数・デットタイムにてパルスを作成し、第五入力の出力許可により１を指定時は出力します。[L なしブロックは第一入力がありません]

図 ２．１３-１２　ADSP324-145 Pwm Output L (3) with Freq

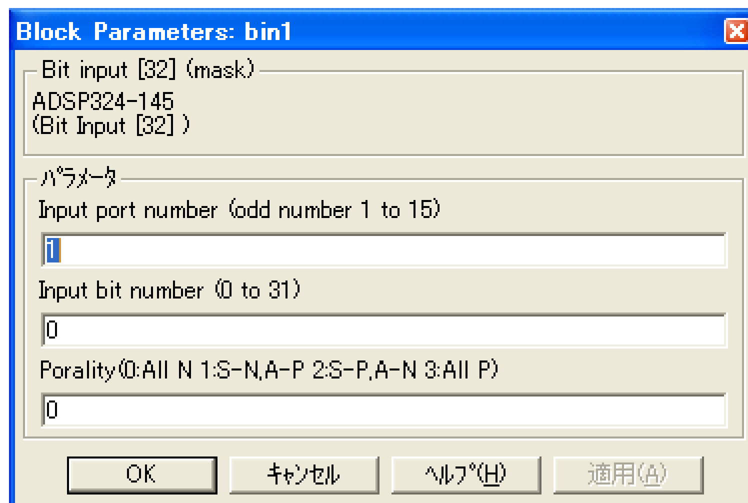


- Channel number (0 to 15)
  ADSP324-145 のPWMチャネル番号を０～１５の整数で指定します。チャネル番号とハードウェアとの対応は、表 ２．１３-３を参照してください。
- Dead Time〔0 to 2000.0 μSec〕
  デットタイムを０～２０００．０μSec にて指定してください。
- Output Mode〔0:full bridge　1:half bridge〕
  波形の出力モードを指定してください。
  ０：フルブリッジ出力 １：ハーフブリッジ出力
- Reflection time〔0:synchronizes cycle 1:time of write〕
  設定値の反映タイミングを指定してください。
  ０：周期に同期（制御周期の切り替わりに同期して設定値を反映)
  １：書きこみ時（モデルの制御周期時に設定値を反映)
- Logic mode〔0:negative 1:positive〕
  同期パルス出力の信号論理を指定してください。
  ０：負論理
  １：正論理

２．１３．１４．PIO bit In〔32〕デバイスドライバブロック

PIO bit In〔32〕デバイスドライバは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは３２ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時 LOW：１・HIGH：０ ・未接続：1,正論理選択時 LOW：０ ・HIGH：１ ・未接続：１です。ボードが接続されていないポートの全てのビットは不定です。

図 ２．１３－１３　ADSP324-145　bit In〔32〕



- Input port number (odd number in 1 to 15)
  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 ２．１３－４を参照してください。

- Input bit number (0 to 31)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 ２．１３－５を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  （TTL レベルのみ有効、絶縁レベル時は指定時も無効）
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．１５．PIO bit In〔16〕デバイスドライバブロック

PIO bit In〔16〕デバイスドライバは、ビット単位入力ドライバです。指定されたポートの指定されたビットの値をスカラーで出力します。このポートは１６ビット単位のアクセスになります。このブロックから出力される値は、ボードへの入力レベルは負論理選択時 LOW：１・HIGH：0 ・未接続：1,正論理選択時 LOW：0 ・HIGH：1 ・未接続：１です。ボードが接続されていないポートの全てのビットは不定です。

図 ２．１３−１４　ADSP324-145　bit In〔16〕

Block Parameters: bin2

Bit input [16] (mask)

ADSP324-145
(Bit Input [16])

ﾊﾟﾗﾒｰﾀ

Input port number (odd number 1 to 15)

1

High and Low Rank (0:High 1:Low)

0

Input bit number (0 to 15)

0

Porality(0:All N 1:S-N,A-P 2:S-P,A-N 3:All P)

0

OK　ｷｬﾝｾﾙ　ﾍﾙﾌﾟ(H)　適用(A)

- Input port number (odd number in 1 to 15)
  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 ２．１３−４を参照してください。

- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Input bit number(0 to 15)
  入力の対象となるビット番号を指定します。ビット番号とハードウェアの対応表は、表 ２．１３−５を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．１６．PIO paralell in〔32〕デバイスドライバブロック

PIO paralell in〔32〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの全ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 ２．１３－１５　ADSP324-145　paralell in〔32〕



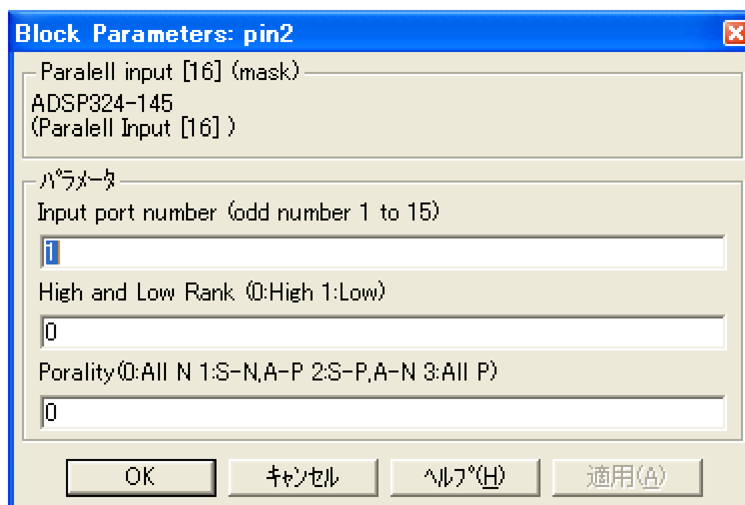
- Input port number (odd number in 1 to 15)
  入力の対象となるビットが含まれるポートの番号を指定します。ポート番号とハードウェアの対応表は、表 ２．１３－４を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．１７．　PIO paralell in〔16〕デバイスドライバブロック

PIO paralell in〔16〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは､幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図 ２．１３－１６　ADSP324-145　paralell in〔16〕



- Input port number (odd number in 1 to 15)
  入力の対象となるビットが含まれるポートの番号を指定します｡ポート番号とハードウェアの対応表は、表 ２．１３－４を参照してください。

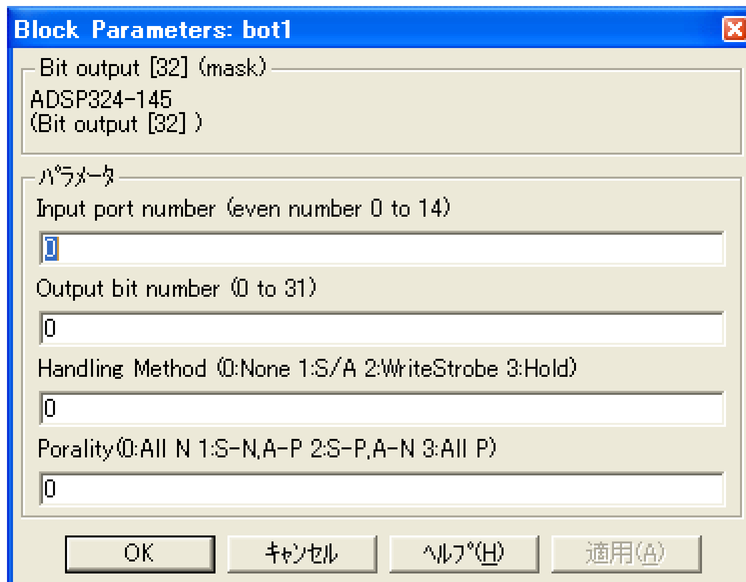
- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．１８．　PIO bit out〔32〕デバイスドライバブロック

PIO bit out〔32〕デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは３２ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが０の時 OFF、１の時 ON です。ボードの出力初期値は全ビット共 OFF です。

図　２．１３−１７　ADSP324-145　bit out〔32〕

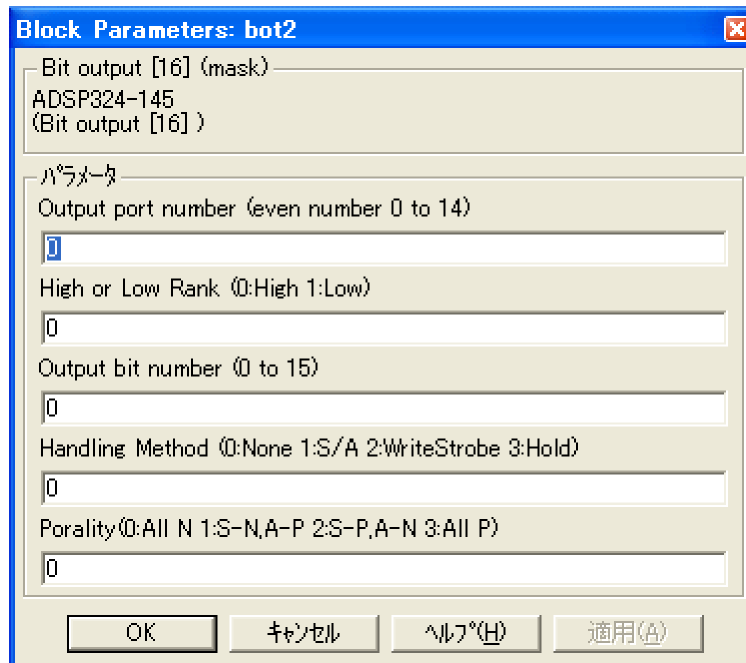


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３−４を参照してください。
- Output bit number〔0 to 31〕
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表　２．１３−５を参照してください。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。
  ハンドリング方式の詳細は、表　２．１３−６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB, ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB, ACK 共正論理

２．１３．１９．　PIO bit out〔16〕デバイスドライバブロック

PIO bit out〔16〕デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは１６ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが０の時 OFF、１の時 ON です。ボードの出力初期値は全ビット共 OFF です。

図　２．１３－１８　ADSP324-145　bit out〔16〕

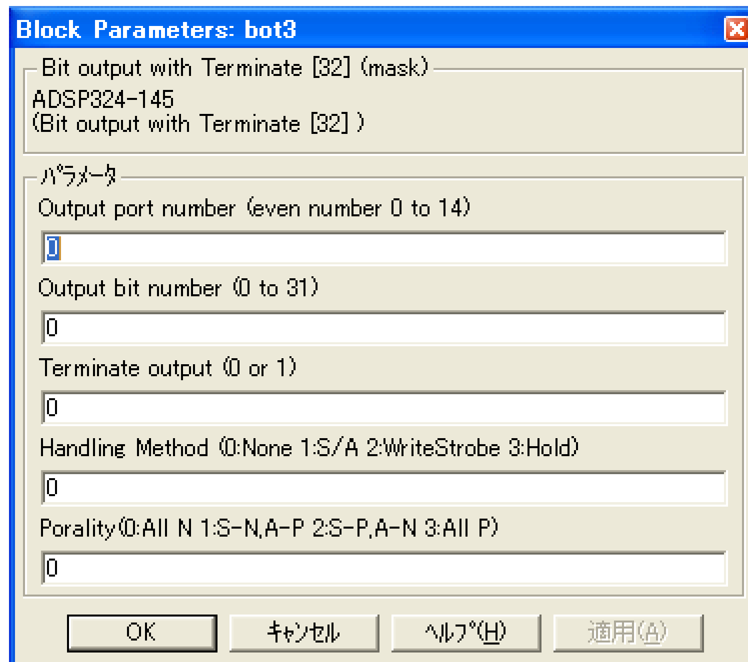


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３－４を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。
- Output bit number〔0 to 15〕
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～１５です。ビット番号とハードウェアとの対応は、表　２．１３－５を参照してください。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は､指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は､表　２．１３－６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．２０．　PIO bit out with Terminate〔32〕デバイスドライバブロック

PIO bit output Terminate out〔32〕デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートのビットへ出力します。このポートは３２ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが０の時 OFF、１の時 ON です。ボードの出力初期値は全ビット共 OFF です。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図　２．１３－１９　ADSP324-145　bit output Terminate out〔32〕

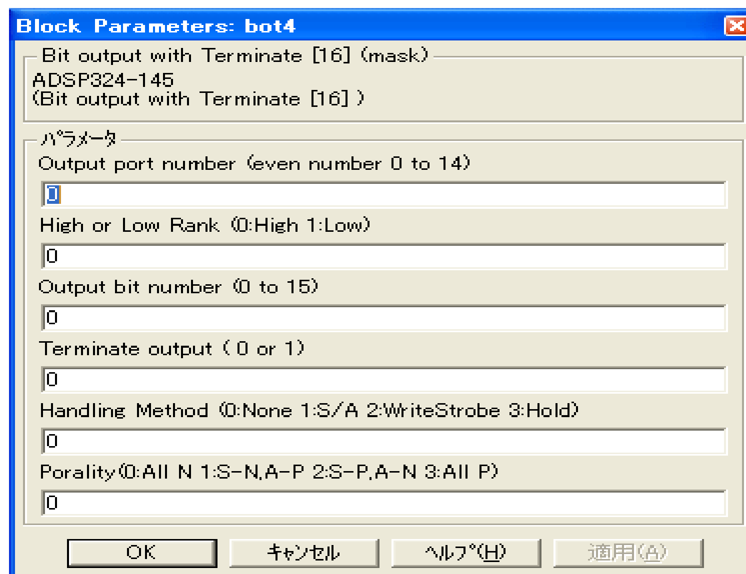


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３－４を参照してください。
- Output bit number〔0 to 31〕
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表　２．１３－５を参照してください。
- Treminate output〔0 or 1〕
  実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表　２．１３－６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  （TTL レベルのみ有効、絶縁レベル時は指定時も無効）
  ０：STB, ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB　正論理、ACK　負論理　　３：STB, ACK 共正論理

2．１３．２１．　PIO bit out with Terminate〔16〕デバイスドライバブロック

PIO bit output Terminate out〔16〕デバイスドライバは、ビット単位の出力デバイスドライバブロックです。スカラーの入力を受け取り、指定されたポートの指定ビット位置（上位 / 下位）へ出力します。このポートは１６ビット単位のアクセスになります。ボードから出力される信号は、このブロックへの入力レベルが０の時 OFF、１の時 ON です。ボードの出力初期値は全ビット共 OFF です。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2．１３-２０　ADSP324-145　bit output Terminate out〔16〕

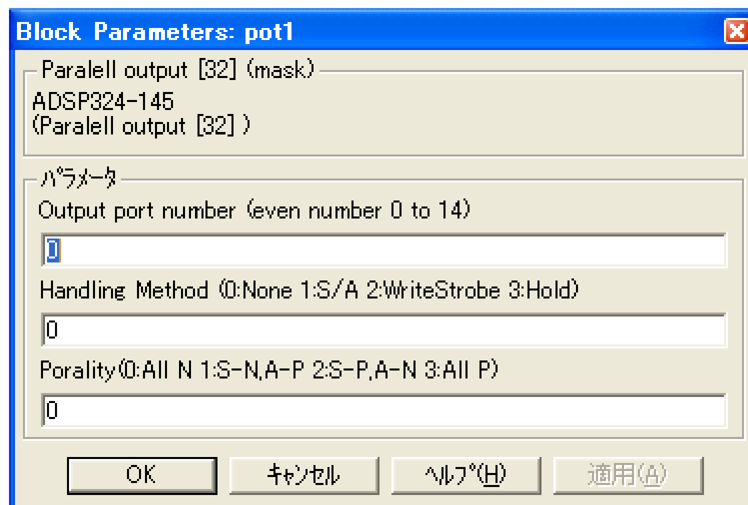


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2．１３-４を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。
- Output bit number〔0 to 31〕
  出力の対象となるビット番号を指定します。指定できるビット番号は、０～３１です。ビット番号とハードウェアとの対応は、表 2．１３-５を参照してください。
- Treminate output〔0 or 1〕
  実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。ハンドリング方式の詳細は、表 2．１３-６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB, ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB, ACK 共正論理

２．１３．２２．　PIO paralell out〔32〕デバイスドライバブロック

PIO paralell out〔32〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの３２ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図　２．１３－２１　ADSP324-145　paralell　out〔32〕



- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３－４を参照してください。

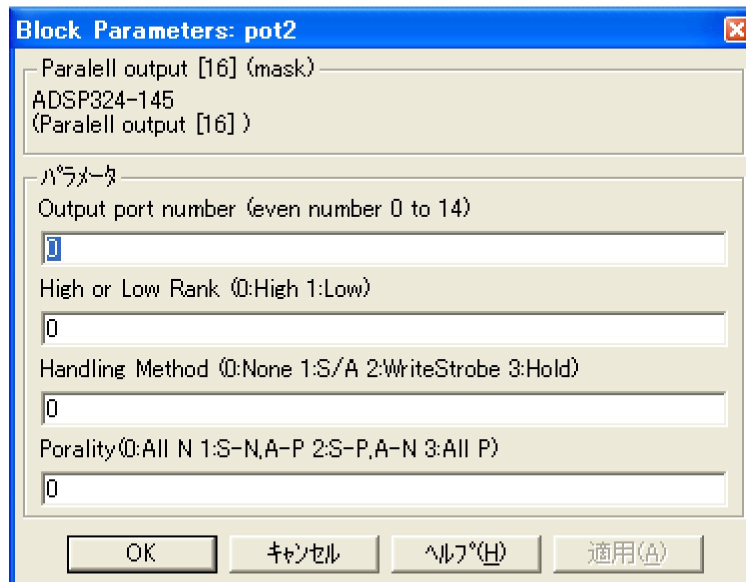
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。
  ハンドリング方式の詳細は、表　２．１３－６を参照してください。

- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  （ＴＴＬレベルのみ有効、絶縁レベル時は指定時も無効）
  ０：STB, ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB　正論理、ACK　負論理　　３：STB, ACK 共正論理

２．１３．２３．　PIO paralell out〔16〕デバイスドライバブロック

PIO paralell out〔16〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは、幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。

図　２．１３－２２　ADSP324-145　paralell out〔16〕

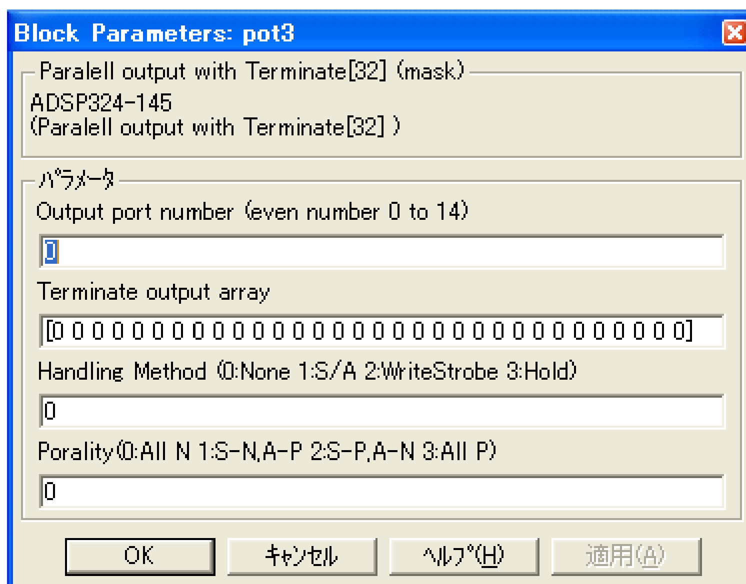


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３－４を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。
  ハンドリング方式の詳細は、表　２．１３－６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．２４．　PIO parallel out with Terminate〔32〕デバイスドライバブロック

PIO paralell Terminate out〔32〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの３２ビットを入力し、結果をベクトルで出力します。このポートは３２ビット単位のアクセスになります。また出力されるベクトルは、幅が３２のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット３１に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。
このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図　２．１３－２３　ADSP324-145　paralell Terminate out〔32〕

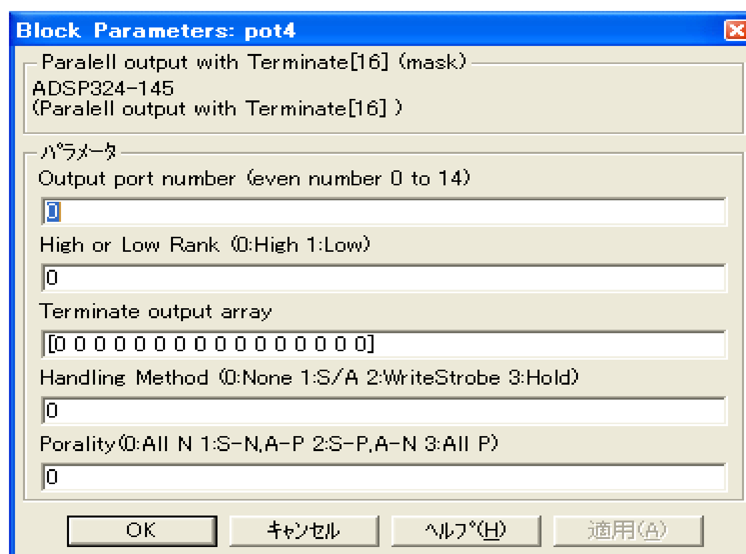


- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表　２．１３－４を参照してください。
- Treminate output〔0 or 1〕
  実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。
  ハンドリング方式の詳細は、表　２．１３－６を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  （TTL レベルのみ有効、絶縁レベル時は指定時も無効）
  ０：STB,ACK 共負論理　　　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　　３：STB,ACK 共正論理

2．1 3．2 5．　PIO paralell out with Terminate〔16〕デバイスドライバブロック

PIO paralell Terminate out〔16〕デバイスドライバは、複数ビット同時入力デバイスドライバブロックです。指定されたポートの指定位置（上位/下位）１６ビットを入力し、結果をベクトルで出力します。このポートは１６ビット単位のアクセスになります。また出力されるベクトルは、幅が１６のベクトルで、先頭の要素がビット０、次がビット１の順で、最後の要素がビット１５に対応しています。各要素はボードへの入力信号が OFF の時０、ON の時１です。入力信号が接続されていないビットは０になります。ボードが実装されていないポートの全てのビットは不定です。このブロックは、実時間モデルが終了する際に予め設定しておいた値を出力することができます。

図 2．1 3−2 4　ADSP324-145　paralell Terminate out〔16〕



- Output port number (even number in 0 to 14)
  出力の対象となるビットが含まれるポートのポート番号を指定します。ポート番号とハードウェアとの対応は、表 2．1 3−4を参照してください。
- Higher and Low Rank(0: Higer 1: Low)
  ポートでの 16 ビットの位置を０：上位又は１：下位で指定してください。
- Treminate output〔0 or 1〕
  実時間モデルが終了時する際に出力したい値を設定します。
- Handling Method〔0: None 1: Hold 2: S/A 3: WriteStrobe〕
  出力のハンドリング方式を指定します。ポート番号毎に決定されます。このブロックを複数使用する場合は、指定ポート番号毎の設定は同一にしてください。
  ハンドリング方式の詳細は、表 2．1 3−6を参照してください。
- Porality (0:All Negative 1:ACK=N STB=P 2:ACK=P STB=N 3: All Positive)
  入力信号の極性を指定します。
  (TTL レベルのみ有効、絶縁レベル時は指定時も無効)
  ０：STB,ACK 共負論理　　　　１：STB 負論理、ACK 正論理
  ２：STB 正論理、ACK 負論理　　３：STB,ACK 共正論理

２．１３．２６．　From GOT デバイスドライバブロック

From GOT デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図 ２．１３-２５　ADSP324-145　From GOT



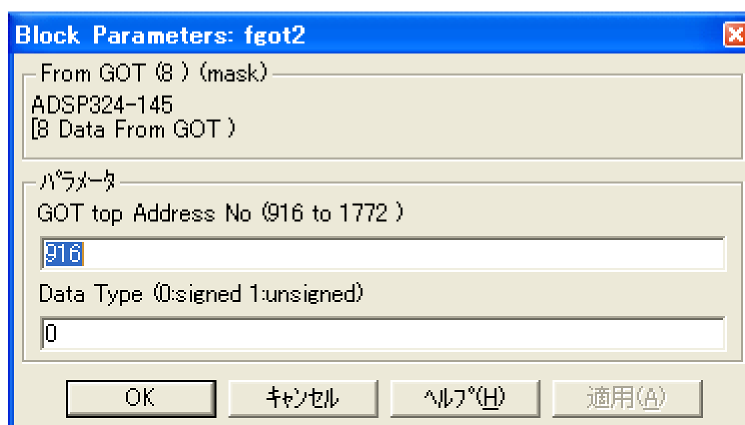
- GOT Address No (916 to 1779)
  GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については　“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．２７．　From GOT[8]デバイスドライバブロック

From GOT[8]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で取得します。

図 ２．１３－２６　ADSP324-145　From GOT[8]



- GOT Address No (916 to 1772)
  GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０： 符号付データ、１： 符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．２８．　From GOT[16]デバイスドライバブロック

From GOT[16]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で取得します。

図 ２．１３-２７　ADSP324-145　From GOT[16]



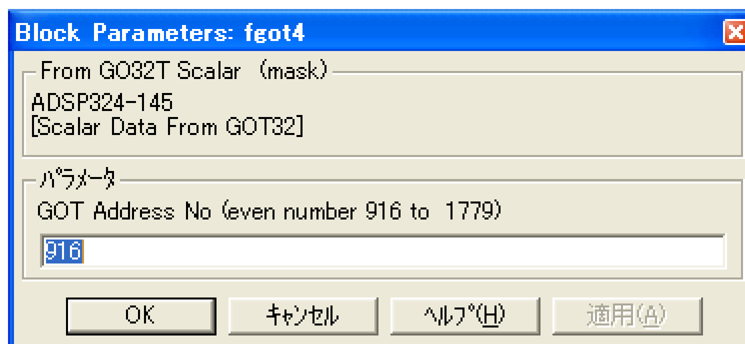
- GOT Address No (916 to 1764)
  GOT の先頭アドレス番号を 916～1764 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０： 符号付データ、１： 符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．２９．　From GOT32 デバイスドライバブロック

From GOT32 デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで取得します。

図　２．１３－２８　ADSP324-145　From GOT32

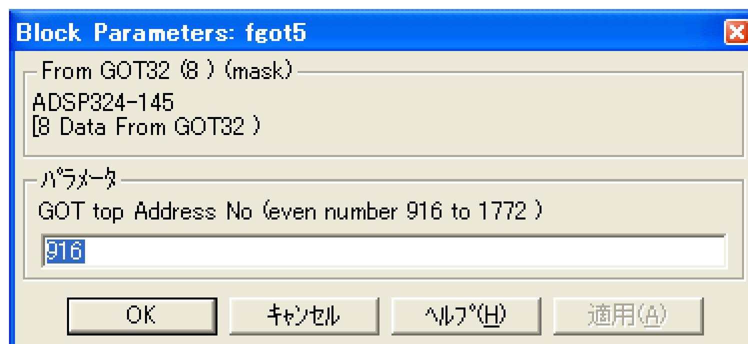


- GOT Address No (916 to 1779)
  GOT のアドレス番号を 916～1779 の範囲で指定してください。詳細については　“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１３．３０．　From GOT32 [8]デバイスドライバブロック

From GOT32 [8]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で取得します。

図 ２．１３−２９　ADSP324-145　From GOT32 [8]

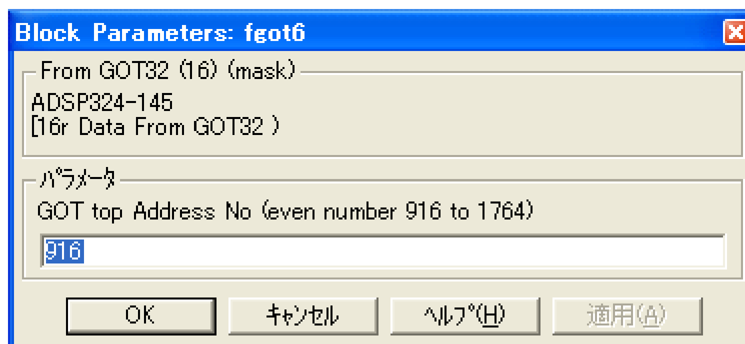


- GOT Address No (916 to 1772)
  GOT の先頭アドレス番号を 916～1772 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１３．３１．　From GOT32 [16]デバイスドライバブロック

From GOT32 [16]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で取得します。

図 ２．１３−３０　ADSP324-145　From GOT32 [16]



- GOT Address No (916 to 1764)
  GOT の先頭アドレス番号を 916〜1764 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１３．３２．　To GOT デバイスドライバブロック

To GOT デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 ２．１３－３１　ADSP324-145　To GOT



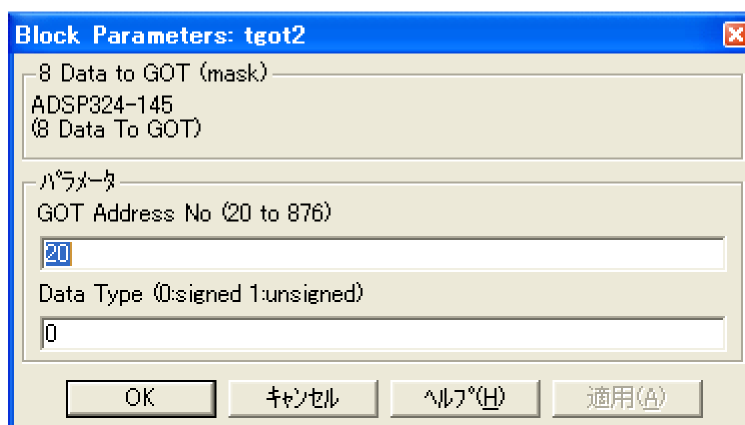
- GOT Address No (20 to 883)
  GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．３３．　To GOT[8]デバイスドライバブロック

To GOT[8]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で出力します。

図　２．１３－３２　ADSP324-145　To GOT[8]



- GOT Address No (20 to876)
  GOT の先頭アドレス番号を 20～876 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．３４．　To GOT[16]デバイスドライバブロック

To GOT[16]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で出力します。

図 ２．１３−３３　ADSP324-145　To GOT[16]



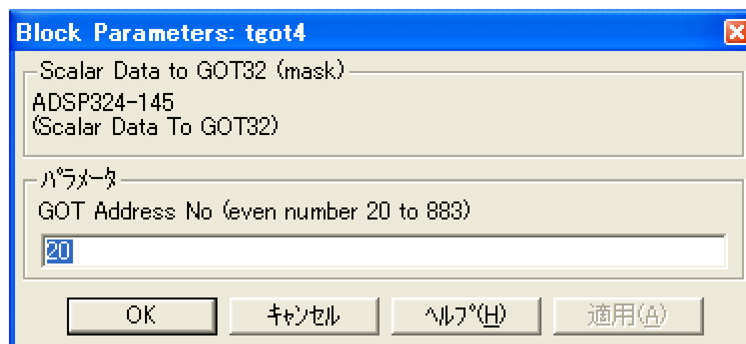
- GOT Address No (20 to 868)
  GOT の先頭アドレス番号を 20～868 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

- Data Type (0:signed 1:unsigned)
  ０：符号付データ、１：符号なしデータ　のいずれかを指定してください。この指定は“ADSP674 操作ターミナル”の画面オブジェクト基本設定値と同一指定にしてください。

２．１３．３５．　To GOT32 デバイスドライバブロック

To GOT32 デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをスカラーで出力します。

図 ２．１３-３４　ADSP324-145　To GOT32

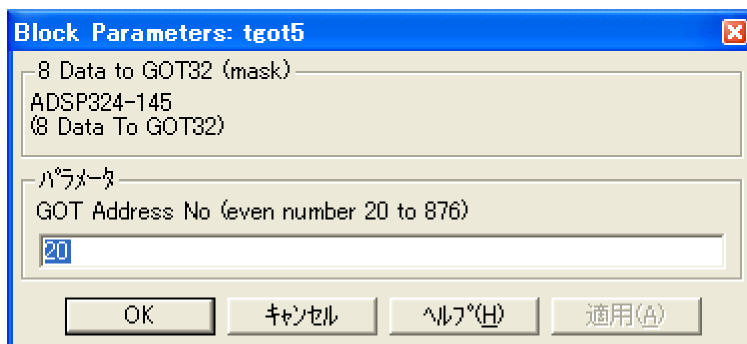


- GOT Address No (20 to 883)
  GOT の先頭アドレス番号を 20～883 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１３．３６．　To GOT32 [8]デバイスドライバブロック

To GOT32 [8]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅８で出力します。

図 ２．１３-３５　ADSP324-145　To GOT32 [8]

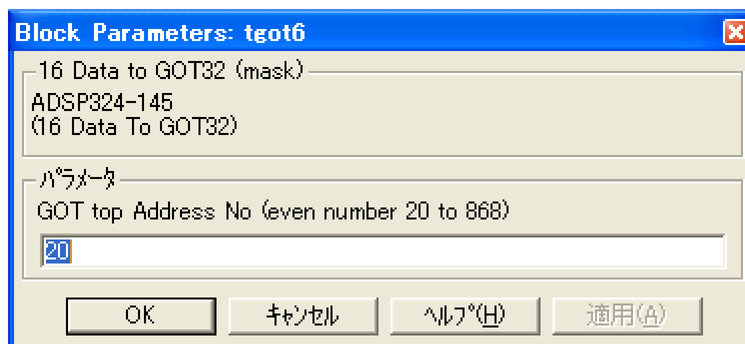


- GOT Address No (20 to876)
  GOT の先頭アドレス番号を 20～876 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１３．３７．　To GOT32 [16]デバイスドライバブロック

To GOT32 [16]デバイスドライバは、ADSP674 操作ターミナル(ADSP-GOT シリーズ)の付属 GOT 用のデバイスドライバブロックです。指定された GOT アドレスのデータをベクトル幅１６で出力します。

図　２．１３-３６　ADSP324-145　To GOT32 [16]



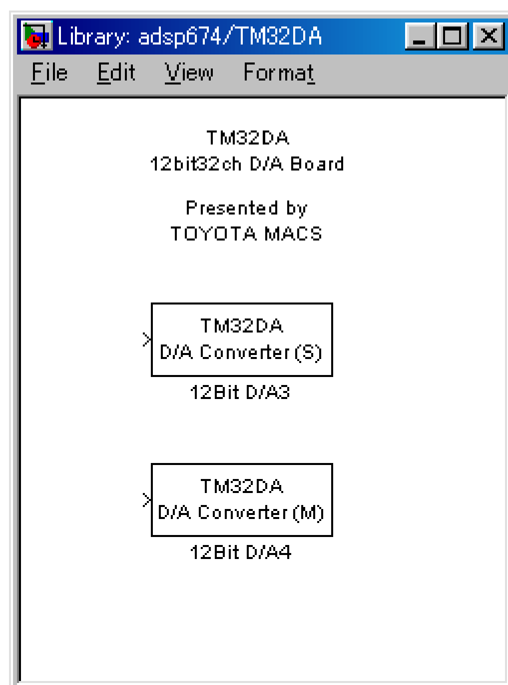
- GOT Address No (20 to 868)
  GOT の先頭アドレス番号を 20～868 の範囲で指定してください。詳細については“ADSP674 操作ターミナル・GOT 操作ガイド”を参照してください。

２．１４．　TM32DA デバイスドライバブロック

２．１４．１．概要

図　２．１４−１に、TM32DA（１２ビット多チャンネルD/Aボード）用デバイスドライバウィンドウを開いた所を示します。

図　２．１４−１　TM32DA ドライバウィンドウ



この中には ADSP324-03 とのインターフェースを取るための４つのデバイスドライバブロックが登録されています。

- D/A input (S)　１チャンネルの D/A 出力デバイスドライバブロックです。
- D/A input (M)　多チャンネルの D/A 出力デバイスドライバブロックです。

１チャンネルの入力（出力）デバイスドライバは、任意のチャンネルからデータを入力（出力）できます。特定のチャンネルのみをアクセスしたい場合や、アクセスしたいチャンネルが連続するチャンネルでない場合に有用です。

多チャンネルの入力（出力）デバイスドライバは、チャンネル番号０から連続する任意の数のチャンネルからデータを入力（出力）できます。複数のチャンネルをベクトルとして取り扱う場合や、入力（出力）の為のオーバーヘッドを低減する目的等で連続するチャンネルを一括してアクセスする場合等に利用します。

１チャンネル版と多チャンネル版は一度に取り扱うチャンネルの数やチャンネル番号の指定が異なるのみで、本質的には全く差異はありません。例えば、多チャンネル版で入力した複数チャンネルの A/D 入力データを Demux ブロックで個別の信号に分解したものと、１チャンネル版で個々に入力したものとは全く同一として取り扱えます。D/A の場合も同様です。特に、A/D では、同じサンプリング時間を設定した複数のデバイスドライバを使用した場合のサンプリングの同時性は保証されています。１チャンネル版と多チャンネル版のドライバを混在した場合でも同様です。

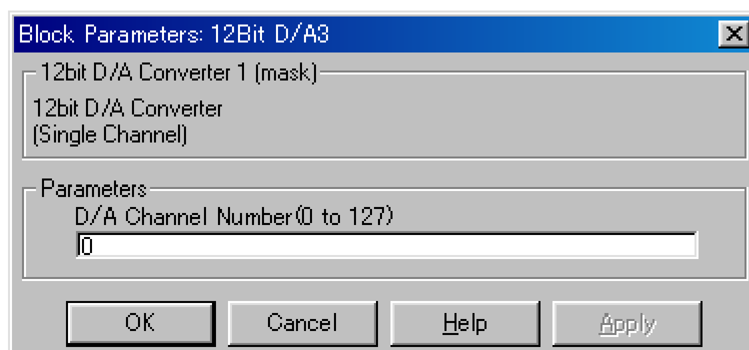
表 ２．１４-１　チャンネル番号とＴＭ３２ＤＡボードとの対応

ボード	CH00	CH01	…	CH30	CH31
１台目	０	１	…	３０	３１
２台目	３２	３３	…	６２	６３
３台目	６４	６５	…	９４	９５
４台目	９６	９７	…	１２６	１２７

２．１４．２．　１チャンネル D/A 出力デバイスドライバブロック

１チャンネル D/A 出力デバイスドライバブロックは、スカラーデータを受取り、指定されたチャンネル番号の D/A へ出力します。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図 ２．１４−２　TM32DA D/A output (S)

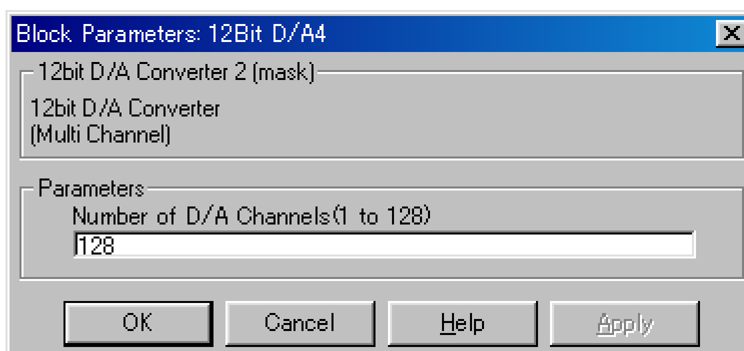


- D/A channel number (0 to 127)
  出力するD/Aのチャンネル番号を０～１２７の整数で指定します。チャンネル番号とハードウェアとの対応は、表 ２．１４−１ (163ページ) を参照してください。

２．１４．３．　多チャンネル D/A 出力デバイスドライバブロック

多チャンネル D/A 出力デバイスドライバブロックは、ベクトルデータを受取り、複数チャンネルの D/A へ一括して出力します。出力の対象となる D/A チャンネルは、チャンネル番号０から連続するチャンネルで、先頭チャンネル番号は０で固定となっています。このブロックへの入力ベクトルの幅と、このブロックに指定したチャンネル数とは一致していなければいけません。このブロックへ入力されるベクトルの、最初の要素がチャンネル番号０の D/A へ出力されます。
このブロックへの入力データの単位は電圧値[V]です。物理単位からの変換は Gain ブロック等を使って予め行ってください。

図　２．１４－３　TM32DA D/A output　(M)



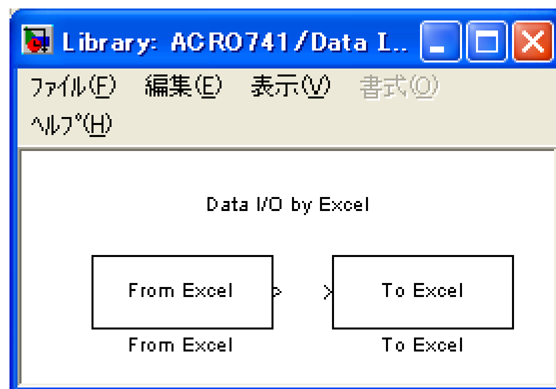
- Number of D/A channels (1 to 128)
  出力するD/Aのチャンネル数を１～１２８の整数で指定します。チャンネル番号とハードウェアとの対応は、表　２．１４－１　(163ページ) を参照してください。

２．１５．　Data I/O By Excel ブロック

２．１５．１．概要

図　２．１５−１に、Data I/O By Excel用ウィンドウを開いた所を示します。

図　２．１５−１　Data I/O By Excel ウィンドウ



この中には自動テスト機能で使用可能な From Excel/To Excel ブロックが登録されています。

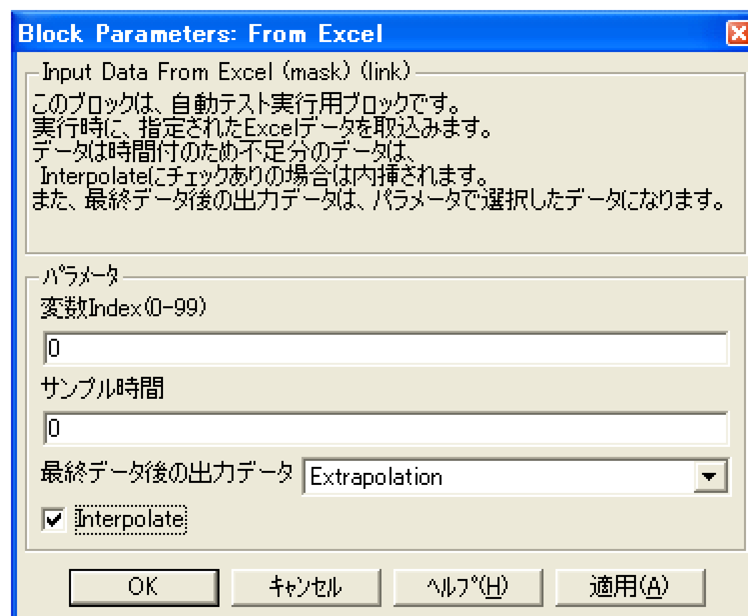
- From Excel　　MicroSoft Excel にて作成したデータを実行時動的にブロックへ取り込みます。
- To Excel　　　MicroSoft Excel ブックへ実行時に動的にデータを出力します。

To Excel ブロックは、自動テストにて指定項目を設定し、実行した場合のみ有効となります。
これらのブロックは、データ用の Excel ブックが必要となります。
１）自動テスト時は、第７章自動テストを参照してください。
２）From Excel を RTMON で使用する場合も Excel ブックが必要となります。Excel ﾌﾞｯｸの作成方法については第７章自動テストを参照してください。RTMON を実行時は“リアルタイムオプション”ダイアログにて Excel ブックの指定と FromExcel ブロック使用にチェックをしてからモデルを実行してください。

２．１５．２．　From Excel ブロック

From Excel ブロックは、Microsoft 社製 Excel にて作成した、時間付データを実行時に動的にアクセスしモデルへデータを取り込みます。このブロックは、モデル実行前にデータを@ＣＲＯへ転送しますので、予め必要な領域をモデル名.lcf にて指定してください。詳細は、**エラー! 参照元が見つかりません。**を参照してください。

図 ２．１５－１　From Excel



- 変数 Index(0-99)
  自動テスト用の Excel ブックで指定する変数 Index を０～９９の範囲で指定します。ToExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。
- サンプル時間
  データを取り込むサンプリングタイムを指定します。０指定時はモデルのステップサイズに依存します。
- 最終データ後の出力データ
  指定した時間以降のデータ実装方法を指定します。
  Extrapolation：直前のデータ内容よりデータを外挿します。
  Setting To Zero:ゼロデータを出力します。
  Holding FinalValue:指定最終データを出力します。
- Interpolate
  データの指定範囲内でモデルのステップタイムに該当するデータがない場合の実装方法を指定します。
  チェックあり：データは内挿されます。
  チェックなし：取り込んだデータの最近のデータ値を出力します。

＜データ取り込み例＞

1）　最終データ後の出力データ: Extrapolation、Interpolate ： チェックあり 指定時

① Excel データ)

time	1	2	3	5	8
data	0	1	5	50	100

↓

ステップサイズ 1 Sec での取り込みデータ)

time	0	1	2	3	4	5	6	7	8	9	10	11
data	-1	0	1	5	27.5	50	66.66	83.33	100	116.66	133.33	150

補間　　内挿　　内挿　　外挿

② Excel データ)

time	0	1	2	2	3
data	0	0	1	5	10

↓

ステップサイズ 1 Sec での取り込みデータ)

time	0	1	2	3	4
data	0	0	5	10	15

＊Excel データが同じタイムステップで複数存在する時は、最後のデータが使用されます。
＊最終データが同じタイムステップで複数存在する時は Extrapolation が保障されませんので、注意してください。

2）最終データ後の出力データ: Setting To Zero、Interpolate ： チェックなし 指定時

① Excel データ)

time	1	2	3	5	8
data	0	1	5	50	100

↓

ステップサイズ 1 Sec での取り込みデータ)

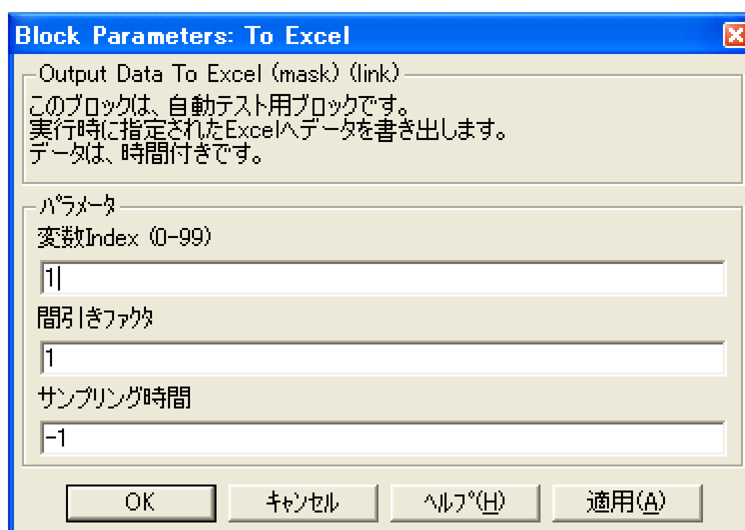
time	0	1	2	3	4	5	6	7	8	9	10	11
data	0	0	1	5	5	50	50	50	100	0	0	0

Setting To Zero

２．１５．３．　To Excel ブロック

To Excel ブロックは、指定された Microsoft 社製 Excel ブックのシートへ、時間と入力データを書き込みます。このブロックは自動テストにてモデルを実行した時のみ有効となります。このブロックは、モデル実行前にデータ領域を＠ＣＲＯ上で確保しますので、予め必要な領域をモデル名.lcf にて指定してください。詳細は、**エラー！ 参照元が見つかりません。**を参照してください。

図　２．１５−２　To Excel



- 変数 Index(0-99)
  自動テスト用の Excel ブックで指定する変数 Index を０～９９の範囲で指定します。FrmExcel ブロックと共通ですので、両ブロック間で重複しないように設定してください。

- 間引きファクタ
  書き込みデータの間引きを指定します。１指定時は間引き無し、２指定時は、２データに対し１データ書込みとなります。

- サンプリング時間
  データを書き込みサンプリングタイムを指定します。

３.異なるＤＳＰ間の互換性

３．１．　異なる DSP 間の互換性

他の異なる環境「ＡＤＳＰ３２４－４１Ｄ」「ＡＤＳＰ６７４－３４１Ｄ」で作成したブロック線図は、一部のブロックを除きコードを再生成する事で他のＤＳＰでの動作が可能です。

図　３．1-1　異種ＤＳＰ用環境間のＩ／Ｏデバイスの互換性（ADSP674-00H/00 の場合）

ブロックライブラリ	互換性
ADSP32X-00/50 ブロック	無し
ADSP32X-03/53 ブロック	有り
ADSP32X-06/56 ブロック	有り
ADSP32X-11/61 ブロック	有り
ADSP32X-13/63 ブロック	有り
ADSP324-141 ブロック	有り
ADSP324-143 ブロック	有り
TM32DA ブロック	有り

４．ボードコントロールツール（ADSP324-00A のみ）

adsp32x ウィンドウの右側寄りにある ＤＳＰ Tools は、ＤＳＰの初期化等を行うツールです。ＤＳＰブロックアイコンをマウス左ボタンでダブルクリックする事によりコマンドを実行します。

４．１． Reset

ＤＳＰボードをリセットするコマンドです。

ＰＣを起動したばかりで、まだ１度もＤＳＰを扱っていない場合、ＤＳＰの内部ＲＡＭは不定となっています。その場合、ＤＳＰをハードウェア的にリセットする目的でこのコマンドを使用します。

又、通常 Real-Time Workshop で作成したプログラムは、（計算結果の妥当性は別として）正常に動作し、暴走等はしない筈ですが、万が一何らかの原因でプログラムが暴走した場合等に、ＤＳＰの状態を初期化する目的でも利用できます。

４．２． Release

ＤＳＰボードを切り離すコマンドです。ＰＣの再起動の前に実行してください。

ＰＣを“Ｗｉｎｄｏｗｓの再起動”や、Ｃｔｒｌ＋Ａｌｔ＋Ｄｅｌ等で再起動する場合、ＤＳＰボードが有効となっていると、ＰＣの機種やメーカーによっては、再起動出来ない場合があります。これは、ＰＣのＢＩＯＳが起動の際にＤＳＰボードのメモリーをＢＩＯＳのＲＯＭの一部と間違えて実行してしまう為です。ＰＣから見たＤＳＰボードはメモリーに見えますが、ＢＩＯＳがＤＳＰである事を知らないが故に、誤判断する為です。

こうした場合ＤＳＰを切り離す必要がありますが、ＤＳＰボードには切り離しの機能が有りません。一旦ＰＣと接続すると接続されっぱなしになってしまいます。そこで止むを得ず、ボード番号１５番のＤＳＰボードを選択する事で機能を代用しています。

ＤＳＰボード番号は標準が０番で、ディップスイッチの変更により０～１５の任意の番号に設定できます。これによりＤＳＰボードを複数接続する場合、重複しないよう設定する事でＤＳＰボードを切り替えてアクセスしています。ある番号のボードが選択されている時、それ以外の番号のボードは切り離された状態になっています。ボード切り離しはこの機能を使い、実装されている公算の最も低い１５番のボードを選択する事により、０～１４番のボードを切り離します。

従って、もし１５番のＤＳＰボードが実装されていると、このコマンドは意図した通り機能しません。又、あえて実装されていないボードを選択する訳ですので、MATLAB Command Window には、“１５番のボードは実装されていない”旨のメッセージが表示されますが、これは異常ではありません。

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2006.09.19	初版

ＭＡＴＬＡＢ対応ライブラリ
Ｉ／Ｏブロック
取扱説明書

中部電機株式会社

〒440-0004　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　FAX <0532>63-1081
URL : 167Hhttp://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006. 9　第１版発行

ＡＤＳＰシリーズ

ユーザ定義ブロック

ＭＡＴＬＡＢ対応ライブラリ

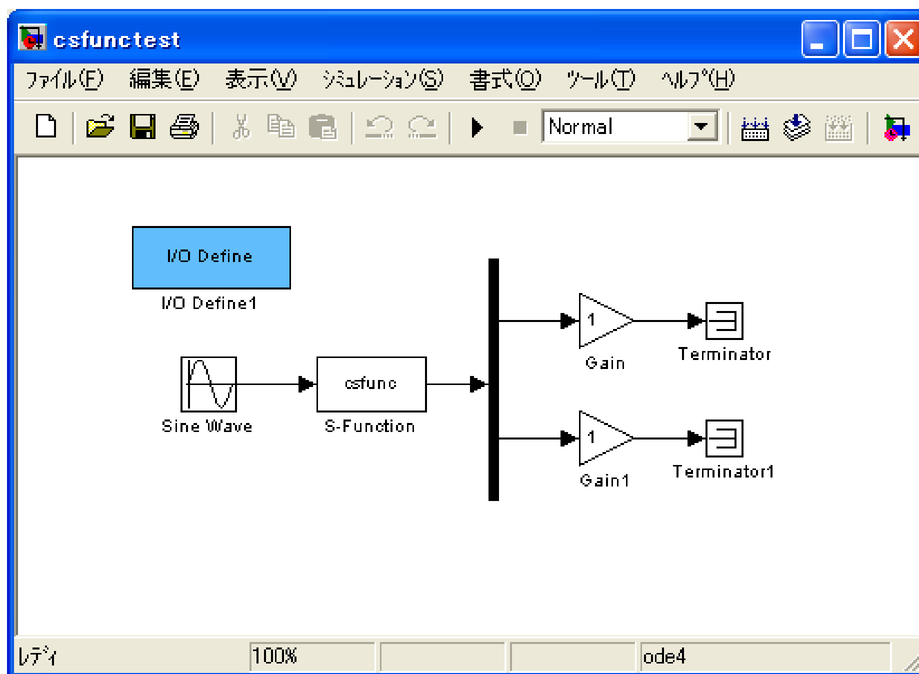
中部電機株式会社

目　次

１． 概要

ここでは、SUMULINK の拡張機能である S-Function を用いて作成されているブロックを用いたブロック線図の実時間コード化について説明します。

S-Function は SUMULINK の拡張機能の一つで、ブロック線図で使用出来る、カスタマイズされたブロックを作成する機能です。出来たブロックは下のブロック線図の様に標準のブロックと接続して、ブロック線図の一部として使用します。



２． 利用出来る S-Function

SUMULINK では、色々な言語による S-Function が利用出来ましたが、Real－TimeWorkshop を用いて PowerPC 用の RT モデルを作成する場合には C 言語による S-function しか利用出来ません。

＜S-Function で利用可能な言語＞

記述言語	SUMULINK	Real-Time Workshop
MATLAB 言語(M ﾌｧｲﾙ)	可	不可
C 言語	可	可
FORTRAN 言語	可	不可

これは専ら、PowerPC 用のコンパイラが、C コンパイラしか無い事に由来しています。
従って、将来 Real-Time Workshop を用いて DSP 用実時間コードを作成する予定がある場合、S-Function は C 言語で作成する必要があります。

３．　ユーザ定義ブロックの作成

３．１．　概略手順

弊社製 DSP-00H/00,ADSP324-00A（以下 DSP）を用いた実時間モデルで使える S-Function によるユーザ定義ブロックの作成手順は以下の通りとなります。

１、ソースファイルの作成
２、DLL の作成
３、SUMULINK での動作確認
４、ライブラリの作成
５、RTW でのコード生成
６、RT モデルでの動作確認

３．２．　ソースファイルの作成

ソースファイルは MATLAB エディタ（又は汎用のテキストエディタ）にて作成します。
SUMULINK には、S-Function のサンプルとして CSFUNC が用意されています。本製品ではこれに倣ったサンプル、CSFUNC.C が以下のフォルダに用意されています。

MATLAB¥RTW¥C¥ADSP67x¥sample¥sfunction　　･･･DSP の場合
MATLAB¥RTW¥C¥ADSP32x¥sample¥sfunction　　･･･ADSP324 の場合

これを参考にすると良いでしょう。同じフォルダにある、csfunctest.mdl はこの S-Function を使用するブロック線図の例です。
この関数及びソースファイルの詳しい内容等、及び S-Function の詳しい説明は SUMULINK のユーザーズガイドをご参照ください。

作成する S-Function のソースファイル名には次に示す制約があります。

- 名称の先頭は、英字で始めてください。
  良い例： a3bc.c
  悪い例： 3abc.c　(理由＝数字で始まっている)
- 名称の文字数は、８文字以内としてください。
  良い例：goodname.c
  悪い例：nameislonger.c
- 名称で使う文字は英数字と ＿ 記号のみとしてください。
  良い例：a32_abc.c
  悪い例：a(#1).mdl　(理由＝かっこ、#等の記号を使っている)
- 英字の大文字と小文字は同じものとして取り扱われます。

作成する S-Function は SUMULINK 用の DLL と、DSP 用ライブラリの２つが必要となりますが、場合によっては、(例えば希望とする処理が数値計算等メモリ内のデータ操作だけで終わってしまう様な場合) 両者は、同一のソースファイルを使う事ができます。
しかしそうでなく、例えば DSP の I/O を扱う様な場合は、DLL は（ＰＣ上で動作し、DSP の I/O は直接扱えませんから）ダミーとなり、DSP 用ライブラリになった時点で初めて処理実体が必要となります。この様な場合は、同一ソースファイルでありながら、部分部分で DLL 用にコンパイルしている時と DSP 用にコンパイルしている時と異なる記述が必要となります。そういった場合には、次の条件付きコンパイルを用いると同一ソースファイルとする事が可能です。

```
#ifdef      MATLAB_MEX_FILE
    /* SIMULINK 用 DLL の時に有効となります。 */
#else
    /* DSP 用ライブラリの時に有効となります。 */
#endif
```

３．３．　DLL の作成

MATLAB では、ユーザ定義 S-Function を含むブロック線図から Real-Time Workshop で実時間コードを作成する場合には、必ず、同 S-Function の DLL 版を用意する必要があります。SUMULINK 上では M ファイル版の S-Function でも動作しますが、Real-Time Workshop によるコード生成は、DLL が無いと正しく行えません。
DLL による S-Function は mex コマンドで行います。Mex コマンドついては SUMULINK のユーザーズマニュアルを参照してください。以下が参考になります。

MATLAB Command window から help mex 又は mex -help を実行

３．４．　DLL 作成用コンパイラの動作保証範囲

mex コマンドで使用するコンパイラについて、本製品出荷時の動作保証範囲は、MATLAB のコンパイラ Lcc　のみとなっています。悪しからずご了承願います。

３．５．　SUMULINK での動作確認

DLL による S-Function が出来たら SUMULINK 上で正しく動作するか確認します。作成した DLL はカレントディレクトリ又は MATLAB の PATH の通ったフォルダに格納します。デバッグの段階ではブロック線図と共にカレントディレクトリに置いておき、正常動作が確認できた段階で PATH の通ったフォルダに移動する方が便利です。
ここでは、DSP の I/O に関する確認は行えません。専らブロック線図との I/F 部分と、計算処理部分についてのみ動作確認します。

３．６．　ライブラリの作成

SUMULINK 上で正しく動作する事が確認出来たら、DSP 用ライブラリを作成します。

RT モデル用ライブラリ化は、以下の手順で行います。

（１）　ライブラリ生成用ディレクトリを作成し、ここに MATLAB カレントディレクトリを移動してください。(MATLAB の CD コマンド参照)

（２）　カレントディレクトリに、S-Function のソースファイルをコピーします。カレントディレクトリには、この S-Function のソース以外の C ソース（拡張子 .c ）を格納しないでください。誤ってライブラリ化されてしまいます。

（３）　ライブラリの作成は、ライブラリ作成コマンド genlib674 (genlib32)で行います。
標準的な使い方は、genlib674(genlib32)　-r　-q　です。これにより C ソースがコンパイルされ、opt_lib.a という名称のライブラリが作成され、所定のディレクトリにコピーされます。

（４）　ライブラリを複数作成する場合は、 genlib674(genlib32)　-r　-q　mylib 等とすると、任意の名称のライブラリ mylib.a が作成されます。 但し、a674_rt/a32x_rt 及び libsrc67/libsrc32 は予約となっていてユーザは利用出来ません。-r はライブラリを所定のディレクトリに自動的にコピーする指示です。この指示をしない場合、カレントディレクトリにライブラリが作成されていますので、後でコピーしてください。-q は -r の指示により自動コピーする際に、以前に作成した同名ライブラリが有っても強制的に上書きします。-q が無いと上書きするか否か問い合わせてきます。

（５）　genlib674(genlib32) -help と入力するとオンラインヘルプが参照できます。併せ参照してください。出来たライブラリをコピーする先のディレクトリ名もこれで確認できます。

（６）　mex で S-Function の DLL コード生成が正しく出来ても、genlib674(genlib32)によるコンパイルでエラーが出る場合があります。この理由の殆どは、C コンパイラのエラーチェック仕様の相違によるものです。DSP 用 C コンパイラの方が（必要以上に）型チェックが厳しい為、ＭＥＸの DLL で正常にコンパイル出来て、正常に動作しているにも拘わらず、エラーが出る場合があります。
その最大のものが、real_T＊と double＊の相違です。real_T は#define により double として定義されていますが、DSP 用コンパイラは別物として扱ってしまいます。例えば simstruc.h で、関数プロトタイプ宣言の引数が real_T ＊ と定義されていて、実際の関数の引数が double ＊ となっていると、エラーとなってしまう場合があります。
コンパイラが出力するエラーメッセージは MATLAB Command Window に表示されますが、ファイル化されテンポラリフォルダーにも格納されています。ファイルの位置については
３．７．ライブラリ作成時のテンポラリフォルダを参照してください。

（７）　ADSP32X シリーズと、DSP とは同じソースファイルを用いてライブラリ化できますが、ライブラリそのものは共用出来ません。同じフォルダで genlib32 と genlib674 をそれぞれ行い、個別にライブラリの作成を行ってください。この時ライブラリ名称を同じとすると、カレントディレクトリ上でライブラリが上書きされますので、生成する都度、それぞれ所定のフォルダにコピーしてください。

注意）DSP ｼﾘｰｽﾞ の場合上記内容は Small モデルを基準に説明しています。Large モデルで作成時は geblib674b にて作成してください。又モデルによりライブラリが格納されるディレクトリが以下のように異なります。

Small モデル　　MATLAB ﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥lib674
Large モデル　　MATLAB ﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥lib674¥big

３．７．　ライブラリ作成時のテンポラリフォルダ

ライブラリ作成時に作業用フォルダの下に下記フォルダが作成されます。
UserSFunLib_gen_Work
この中には以下のファイルが格納されています。

オブジェクトファイル	*.o	S-Funtcion のソースファイル名の拡張子を.o に変えたファイルです。
エラーログファイル	err	コンパイル中の作業進行状況とその途上で発生したコンパイルエラーが記録されています。コンパイラが出力したメッセージそのものをファイル化したもので、テキストエディタで閲覧が可能です。エラー訂正の為の情報となります。閲覧後に再度ライブラリを作成する場合はエディターを一旦閉じないと最新のエラー情報がファイルに反映されない場合がありますので注意が必要です。
ライブラリファイル	*.a	ユーザが指定した名称の（指定を省略した場合はデフォルトの名称の）ライブラリファイルです。コンパイルエラーが無い場合のみ作成されます。一度正しく作成された後 S-Function のソースファイルを変更し、再度ライブラリ作成をしようとしたときにコンパイルエラーが発生した場合はライブラリは古いままで残っています。手動でコピーする場合は日付を良く確認してください。

これらのファイルは、ライブラリが正常に作成され、かつ、ライブラリを所定のフォルダにコピーした後は不用となります。フォルダごと削除しても構いません。

３．８．　RTW でのコード生成

ライブラリ化が終わり、作成されたライブラリが所定のフォルダに格納された後、再度 Real-Time Workshop 実時間コードを生成します。これにより先程作成されたライブラリは自動的にリンクされます。

３．９．　RTW での動作確認

Real-Time Workshop で生成した実時間モデルは、リアルタイムモニタにて実行してみて、要所要所の波形を見る（或いはロギングして見る）のが唯一の方法です。
DSP 用ライブラリになってからの、S-Function 単体のデバッグは一般には困難です。
もし S-Function（又はその内部）のデバッグを行う必要があるなら、ソースファイルにパッチを当てる事で動作結果をメモリー（大域変数）に残す様に変更し、リアルタイムモニタにて動作後、弊社付属ソフトウェア（DSP ユーティリティ）にてメモリを表示する方法が使えます。メモリアドレスは、モデルと同じディレクトリにマップファイル *model*.map が生成されていますので、これを参照してください。

３．１０．CCS での動作確認

もし CCS と XDS をお持ちであれば、これを用いて DSP ボード上で S-Function のデバッグが可能です。具体的方法はユーザ定義関数と同じですのでそちらを参照してください。

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2006.09.19	初版

MATLAB 対応ライブラリ
ユーザー定義ブロック

中部電機株式会社

〒440-0004　　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006. 9　　第１版発行

ＡＤＳＰシリーズ

ユ ー ザ 定 義 関 数

ＭＡＴＬＡＢ対応ライブラリ

中部電機株式会社

目　次

１． 概要

本章は、MATLAB 対応ライブラリに於いて、リアルタイムモデル（以下 ＲＴモデル）にユーザ定義関数を組み込む方法について説明しています。

この機能はSIMULINKでは記述しづらい手続きや、より高速化／高機能化等を実現したい場合、ユーザのＣ言語によるソフトウェア資産を活用したい場合等に、ユーザが記述した C 言語の関数をSIMULNIKブロック線図で記述した処理に組み込む事により、両者を交互に実行する事により、擬似的に並行して実行する為ものです。予め定められた関数名と大域変数を用いる事によりブロック線図とのインターフェースを取り、データを交換する事もできます。

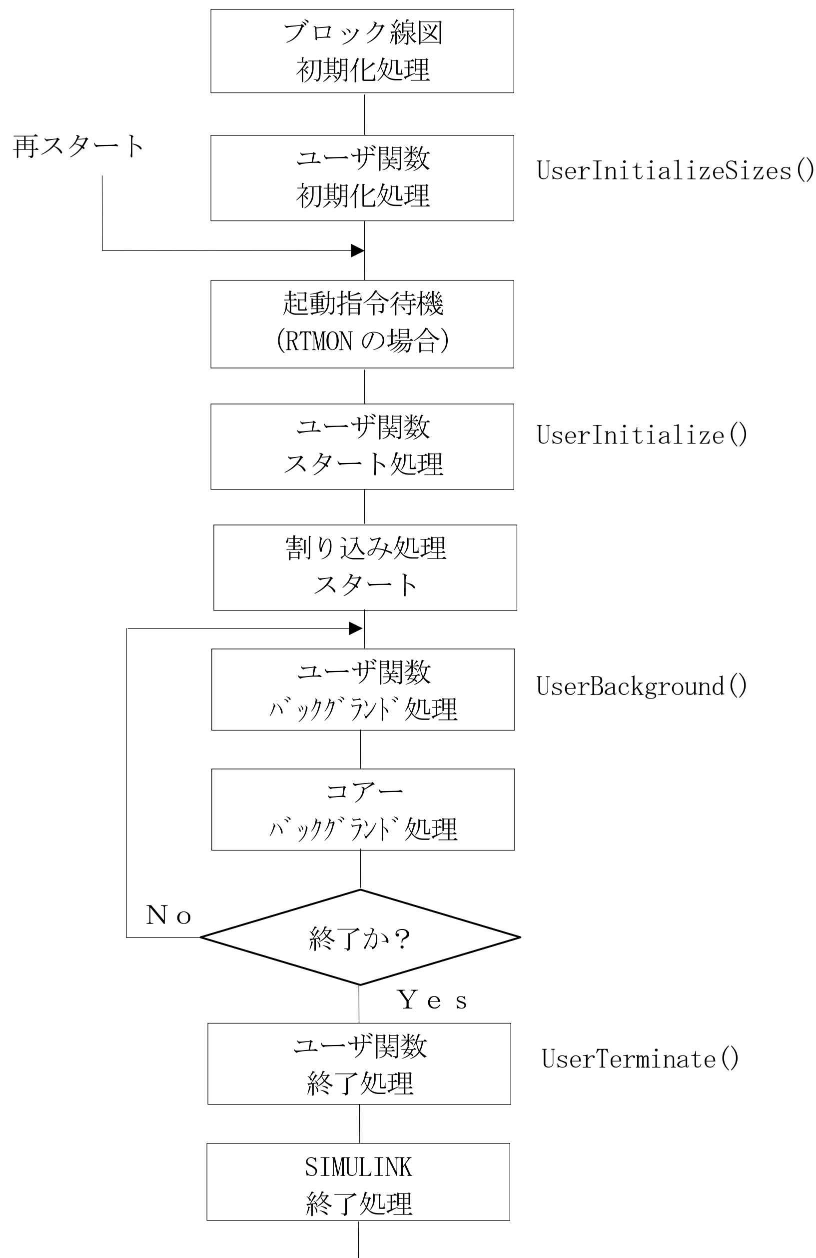
２．実行形態

ここでは、SIMULINK ブロック線図の処理とユーザ定義関数の処理との順序関係、データ交換の方法等を説明します。

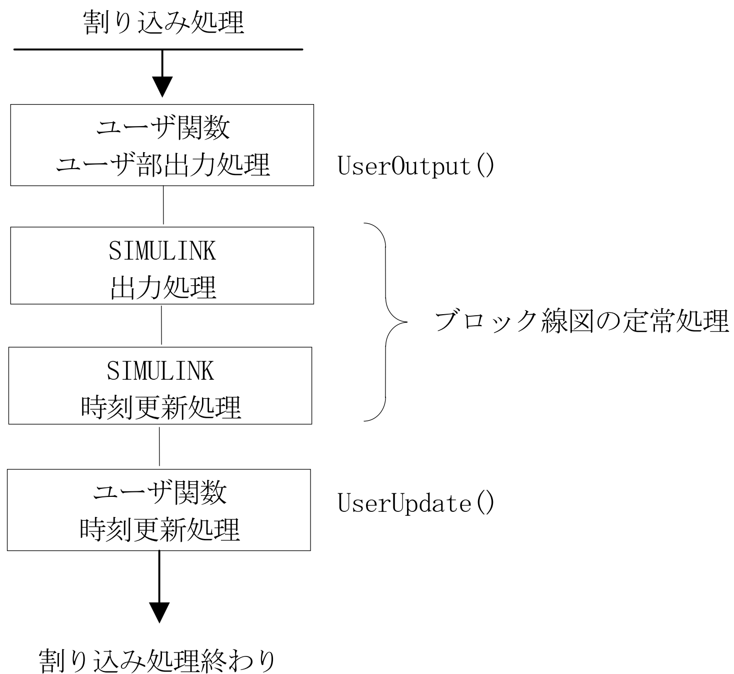
２．１． 処理順序

処理の順番は以下のフローチャートにより行なわれます。

- メインルーチン



● 割り込み処理ルーチン



メインルーチンは、初期化・起動司令待機後、割り込みを起動したらバックグランド処理を行います。ここでは通常、

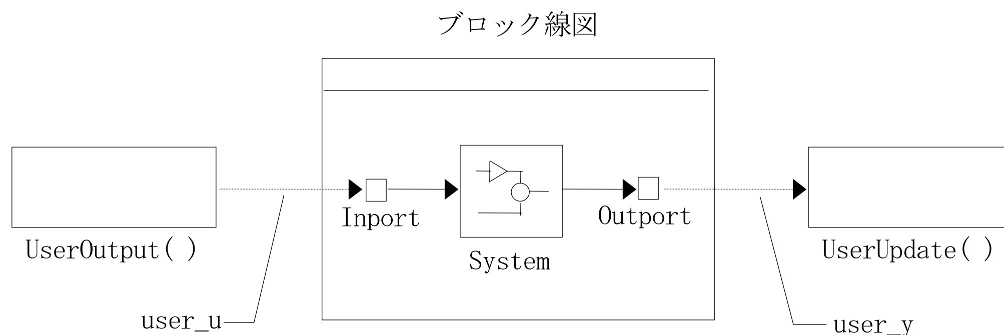
１、 RT モデルが終了時刻に達したか確認を行う。
２、 RTMON、又は、UIL からのランタイムアクセスを処理する。
３、 それ以外のバックグランド処理を行う。

以上の作業を行っています。

一方、割り込み処理は、実行形式によりタイマ又は外部割り込みにより駆動され、RT モデルの１ステップ分の計算を行い、メインルーチンへ戻ります。

フローチャートに示す通り、ユーザ定義関数では、これらの処理の要所々々でユーザの定義する関数を呼び出すことが出来ます。

別の見方として、SIMULINK ブロック線図的な表記では下記の様な接続関係となります。



中央が SIMULINK のブロック線図で、両横の　　　がユーザが定義した処理です。
処理順序は図の向かって左側から右側へ順に処理されます。破線矢印（　　）は大域変数を使用したデータを転送を表しています。ブロック線図には現れません。このように、ユーザ処理と SIMULINK ブロック線図との間のデータ交換は SIMULINK のトップレベルの Inport、Outport を通じて行います。user_u,user_y と Inport,Outport との対応の詳細については“データ交換”の章を参照してください。

３．関数詳細説明

ユーザ定義の関数は予め予約された関数名が６つ用意されています。

UserInitializeSizes()	ユーザ固有の初期化処理を行います。また、ユーザ関数とSIMULINKとのデータ交換に使う領域もここで確保します。
UserInitialize()	システムが機動・停止を繰り返す必用がある場合、再起動に必用な初期化処理を行います。
UserOutput()	ユーザブロックからSIMULINKへの出力を計算します。
UserUpdate()	SIMULINKの出力を受け取り処理すると共に時刻更新の処理を行います。
UserBackground()	ユーザのバックグランド処理を行います。
UserTerminate()	システム停止の処理を行います。

上記関数はテンプレートファイル　USERFUNC.C　に予め定義されており、最低限必要な処理が予め記述されています。ユーザはこれに必用な手続きを書き足します。関数名が定められている事を除けば、通常の関数と同じです。

３．１．　UserInitializeSizes()

関数定義　int　UserInitializeSizes(int in_num, int out_num);

意味　この関数はモデル起動直後に１度だけ呼び出されます。システムの初期化、作業領域の確保等の初期化作業のうち最初に１度だけ行えばよい処理をここで行なってください。

引数　int　in_num;

SIMULINKブロック線図への入力ポート（Inport）の数が与えられます。データ受け渡しに使用する大域変数配列user_uは、最低限これ以上の長さが必用です。この変数は同時に、他のユーザ定義関数から参照可能とする為、u_num等の大域変数に保存する事をお勧めします。これによりuser_uを、確保した長さ以上にアクセスしないよう管理可能です。

int　out_num;

SIMULINKブロック線図の出力ポート（Outport）の数が与えられます。データ受け渡しに使用する大域変数配列user_yは、最低限これ以上の長さが必用です。user_yは、ユーザの使用の有無に関わらず、out_numの点数分書き込みが行われます。この変数は同時に、他のユーザ定義関数から参照可能とする為、y_num等の大域変数に保存する事をお勧めします。これによりuser_yを、確保した長さ以上にアクセスしないよう管理可能です。

戻り値　エラーの有無を返します。正常終了で０を、それ以外は０以外を返します。エラーを返すとメインルーチンは処理を中断します。リアルタイムモニタでは、このエラーをヒープ領域不足として扱い、ステータス表示します。ここでユーザがエラーを返す可能性がある場合、表示の意味に注意してください。

３．２．　UserInitialize()

関数定義　void　UserInitialize(void);

意味　システムが起動停止を繰り返すような場合、起動ごとに必用となる初期化処理をここに記述します。

引数　ありません。

戻り値　ありません。

３．３．　UserOutput()

関数定義　void　UserOutput(double　t);

意味　定常状態でステップサイズ毎に､最初に呼び出される処理です｡ユーザ処理部分の出力、（即ちSIMULINKブロック線図への入力）を決定します。

引数

double　t;

現在の時刻が与えられます｡システム起動後最初の呼び出しには､リアルタイムモニタ、又はSIMULINKのReal-Time Optionsで指示した開始時刻(Start time)が与えられます。通常は０です。その後、呼び出し毎にステップサイズだけ加算されます。

戻り値　ありません。

３．４． UserUpdate()

関数定義　　void　　UserUpdate(void);

意味　　　　定常状態でステップサイズ毎に、SIMULINK ブロック線図の計算が終了した直後に呼び出される処理です。必用に応じてSIMULINK ブロック線図の出力を受け取り、処理をします。また、時刻更新の為の処理もここで行います。

引数　　　　ありません。

戻り値　　　ありません。

３．５． UserBackground()

関数定義　　void　　UserBackground (void);

意味　　　　ユーザのバックグランド処理を記述します。通常メインルーチンでのバックグランド処理は、割り込みによるRTモデルの計算を待ち受ける傍ら、RTMON, UIL等によるランタイムアクセス（実行時のパラメータ変更）をソフト的なハンドリングで行っていますので、これらを妨げないように、時分割で処理する必要があります。長時間かかりそうな処理は、何回かに分け、この関数が呼び出される毎に順番に一部分づつ実行するようにしないといけません。詳しくは、７．３． 長時間に渡るDSPの専有 を参照してください。

引数　　　　ありません。

戻り値　　　ありません。

３．６． UserTerminate()

関数定義　　void　　UserTerminate(void);

意味　　　　システムの終了処理を記述します。RT モデルが停止する直前に１度だけ呼び出されます。RT モデルは以下の条件で停止します。

１、 現在時刻が終了時刻（Stop time）に到達した場合。
２、 ブロック線図の Stop Simulation ブロックの入力が０以外になった場合。
３、 リアルタイムモニタの停止ボタンが押された場合。
４、 ユーザプログラムから、ＵＩＬ経由で RT モデルの停止機能が呼び出された場合。
５、 ユーザ定義関数で、シミュレーション停止フラグを操作した場合。

引数　　　　ありません。
戻り値　　　ありません。

４．データ交換

４．１．　データ交換規則

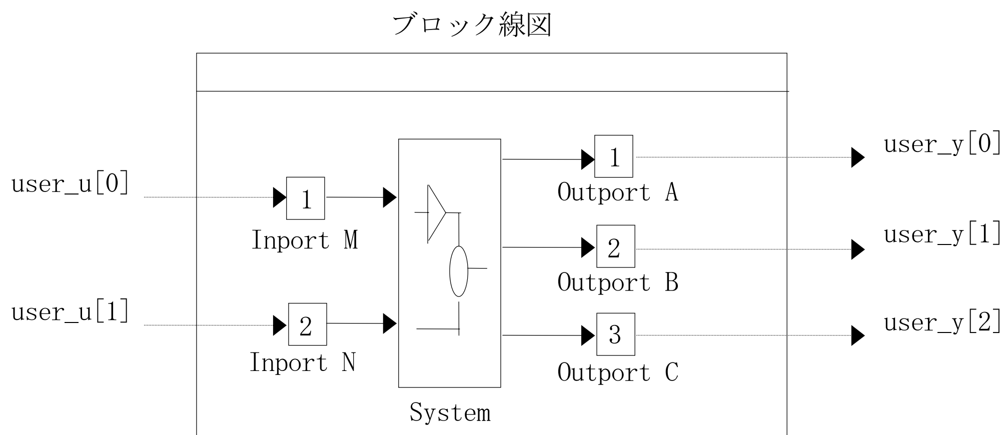
　ユーザ定義処理と SIMULINK ブロック線図との間は大域変数を経由してデータ交換をします。データ領域は USERFUNC.C に大域変数として定義されており、実体は初期化関数 UserInitializeSizes() 内で malloc() にて割り付けます。

double	*user_u;	SIMULINK ブロック線図への入力データを格納します。
double	*user_y;	SIMULINK ブロック線図からの出力データが格納されます。
int	u_num;	user_u の長さを記憶するのに使用します。
int	y_num;	user_y の長さを記憶するのに使用します。

user_u、user_y の長さはそれぞれ u_num、y_num で管理し、確保した長さ以上にアクセスしないように注意してください。 user_u[ ] (user_y[ ]) の添え字の最大は、u_num-1 (y_num-1) となります。

user_u、user_y と SIMULINK ブロック線図の Inport、Outport との対応は下記に示す通りです。この場合は、Inport、Outport はいずれもスカラーとします。

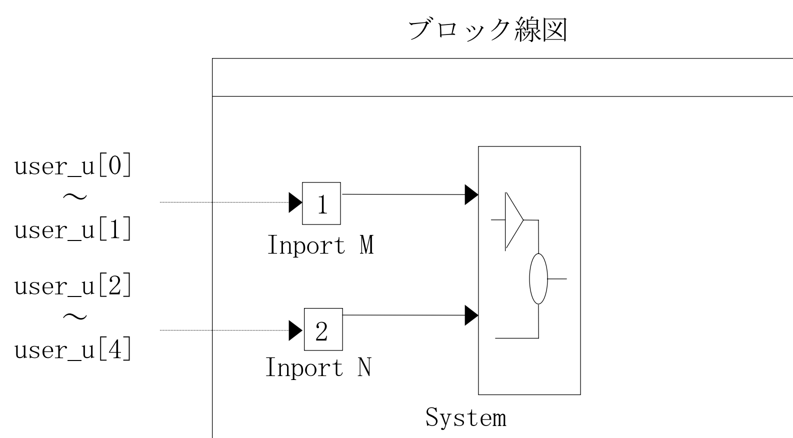
ブロック線図



この例のように、SIMULINK ブロック線図のポート番号（□の中の番号）－１が配列の添え字となります。この場合、u_num == 2、y_num == 3 となります。

Inport、Outport がベクトルの場合の例を次に示します。

ブロック線図



１本のベクトルは配列の中の連続する変数に割り当てられ、ベクトルの要素が若い方が配列の添え字の若い方に相当します。次の Inport はそのすぐ後の変数に相当します。
例えば、このブロック線図で、Inport M が幅２、Inport N が幅３のベクトルであるとすると、以下の様に対応します。

user_u[0]	Input M（１）
user_u[1]	Input M（２）
user_u[2]	Input N（１）
user_u[3]	Input N（２）
user_u[4]	Input N（３）

４．２．　入出力点数の自動整合

ユーザ定義関数でブロック線図との間でデータ交換をする、入出力ブロックの点数（user_u[ ] の添え字）は、ブロック線図とユーザ定義関数とで整合していないといけません。特に、ブロック線図で定義された以上の長さについてアクセスをすると、場合によってはPowerPC が暴走する等の不都合が生じる場合があります。

MATLAB では、この点数を表す定数がヘッダに定義されており、これを参照する事により、コンパイル時点でエラーを発生させてユーザに警告する事が可能です。
例えばユーザ定義関数側が出力（ユーザ出力→ブロック線図）として　２つのポートを、入力（ブロック線図→ユーザ時刻更新）として３つのポートを、それぞれ想定してコードを作成した場合、*MODEL*.UFN コードの適当なヵ所に、

```
#if  ML51
#if    NINPUTS != 2
 #error : Not expected number of InputPort of Model.
#endif
#if NOUTPUTS != 3
 #error : Not expected number of OutputPort of Model.
#endif
#endif
```

と記述します。

NINPUTS がブロック線図への入力、NOUTPUTS がブロック線図からの出力の点数を表しています。点数とはブロックの数でなく、各ブロックのベクトルの、幅の合計ですので注意が必要です。入力ブロックが１個でも、ここへ接続されている信号が幅３のベクトルの場合 NINPUTS は 3 となります。
もし、ブロック線図に想定外の点数の入出力ポートが定義されると、 Not expected number … が MATLAB Command window に表示され、コード生成作業が中断されます。

また、この定数を参照する事により、自動的に入出力点数を整合させる事も可能です。コンパイル時点で点数が判明していますので、作業用変数領域の確保が

```
double  work[NINPUTS];
```

等として的確に確保する事ができます。

４．３．　入出力受け渡しバッファ

user_u、user_y でも受け渡しができますが、入出力ポートを定義した大域変数が用意されており、これを直接参照する事もできます。入力ブロック、出力ブロックを、それぞれ構造体で定義しています。構造体定義は *MODEL*.H にあり、実体は *MODEL*.PRM に記述されています。

＜例：*MODEL*.H （抜粋）＞

```
/*
 * External Inputs Structure
 *
 * Note: Individual field names are derived from the block name.
 *
 */

typedef struct ExternalInputs_tag {
  real_T In[2];
} ExternalInputs;

/*
 * External Outputs Structure
 *
 * Note: Individual field names are derived from the outport block names
 *
 */

typedef struct ExternalOutputs_tag {
  real_T Out[3];
} ExternalOutputs;
```

＜例：*MODEL*.PRM （抜粋）＞

```
/* External Inputs Structure */
ExternalInputs rtU;

/* External Outputs Structure */
ExternalOutputs rtY;
```

５．ユーザ定義関数からの RT モデル停止

ユーザ定義関数から RT モデルを安全に強制停止させるには、以下の方法を用いてください。何れの方法も、停止要求が受理された後、ブロック線図の停止処理、ユーザ定義関数の停止処理等が呼び出され、ＲＴモデルが停止します。
これ以外の方法（例えばタイマを強制的に止める等）で停止させた場合、本来行うべき終了処理（Terminate）が呼びだされません。

５.１． 方法１ ～ Stop Simulation

ユーザ定義関数の出力 user_u を使って、停止制御用信号をブロック線図に渡します。ブロック線図ではこれを Stop Simulation ブロックに入力します。
常時は、０を出力しておき、停止させたい場合は１を渡します。

この方法は、停止する直前にブロック線図でもこれを参照して、何がしかの処理が行えますので、ブロック線図内で停止直前に行うべき処理がある場合に利用できます。

５.２． 方法２ ～ シミュレーション停止フラグの操作

メイン関数に対し、シミュレーションを停止させる要求を渡すフラグが用意されています。
これを直接操作します。

参照変数	extern SimStruct *const rtS;
停止指示	ssSetStopRequested(rtS,1);

この方法はブロック線図に頼らず強制的に停止が可能です。

6．使用方法

6．1． USERFUNC.C のインストール

本製品のインストール直後の状態では、ユーザ定義関数は使用しない状態になっています。これを使用出来るようにするには、以下の作業を行います。

MATLAB をインストールしたフォルダ（%MATLAB_ROOT%）以下の,
次に示すフォルダに USERFUNC.C が格納されています。これを MATLAB のカレントディレクトリにコピーします。

%MATLAB_ROOT%¥rtw¥c¥adsp32x¥rt674(rt)

その後、MATLAB のカレントディレクトリの USERFUNC.C を必要に応じて変更してください。

6．2． USERFUNC.C のコンパイル

6．2．1． 自動によるコンパイル

ユーザ定義関数のファイル“USERFUNC.C”は、上記変更により、Buildボタンから自動的にコンパイルされます。記述したユーザ定義関数にエラーが無ければこの方法で充分なので、特にユーザが個別にコンパイルする必要はありません。
しかし一般には、プログラマーが記述したソースファイルは色々な意味でエラーを含んでおり、書きっぱなしでそのまま正常に動作するのは稀なのが通常です。当然、１つでもエラーがあると実行ファイルは生成されません。その度に USERFUNC.C を修正し再度 Build をするのは、SIMULINK モデルのコード生成とコンパイルをも毎回行う事になり、その分、無駄な時間を費やす事になります。
そこで、最低限タイプミスや文法上の誤りを USERFUNC.C から予め取り除いておく事をお勧めします。

6．3． 個別モデル専用のユーザ定義関数

ユーザ定義関数ソースファイル userfunc.c は、同じディレクトリ内にある全てのモデルで共用となり、このディレクトリ内で作成される全ての実時間モデルに、同じ関数がリンクされます。
もし、モデル毎に異なるユーザ定義関数を用意したい場合は、ソースファイルの名称を標準の userfunc.c でなく、*model*_UFUNC.c とします。（ *model* はブロック線図の名称と同じ）
ファイルの内容は同じで、単にファイル名称を変更するだけです。
同じディレクトリにuserfunc.cと*model*_UFUNC.cとが共に存在する場合は以下の優先順位となります。

＜リンクされるユーザ定義関数＞

順位	条件（ソースファイルの存在）	採用されるソースファイル
1	*Model*_UFUNC.c と userfunc.c が共にある場合。	*Model*_UFUNC.c
2	*Model*_UFUNC.c が無く userfunc.c が有る場合	userfunc.c
3	*Model*_UFUNC.c、 userfunc.c 共に無い場合	無し

７． 制限事項

７．１． 動作上の機能制限

ユーザ関数組み込み機能については、SIMULINK 画面からは利用できません。
SIMULINK 画面でシミュレーションを実行した場合、弊社製品の有無にかかわらず、トップレベルのブロック線図の Inport については常に０が入力され、又、Outport については出力は無視されます。
ユーザ関数とのインターフェースを、SIMULINK 画面で確認するには、Inport の代わりにダミーのサブブロックを、Outport の代わりに Scope や To-Workspace 等を使い、模擬的に行ってください。

７．２． 使用出来ないソースファイル名

ユーザ定義関数では、利用出来るソースファイルの名称に制限があります。本製品で使用しているソースファイル名称と重複すると、不都合が生じますので注意してください。
本製品では、以下のソースファイルを使用しています。尚、版により若干変更がある場合がありますので、念の為、ディレクトリを直接参照し、確認する事をお勧めします。

● ディレクトリ：MATLAB ｲﾝｽﾄｰﾙﾌｫﾙﾀﾞ ¥rtw¥c¥adsp32x¥rt674(rt)

● ヘッダファイル

adsp32x.h	minimon.h
adspio.h	rt5ode.h
adspiost.h	rt5sim.h
adsptype.h	rt5timer.h
cg_sfun.h	rt_error.h
common_145def.h	rt_log.h
ctrl.h	rt_logging.h
dsptype.h	rt_matrx.h
extrnfcn.h	rt_mxclassid.h
logopt0.h	rt_trig.h
logopt.h	rtlibsrc.h
minimon674.h	rtwlog.h
	simstruc.h
	simstruc_types.h
	tmwtypes.h

●ソースファイル

timsub6x.c(ADSP674 のみ)	rt5sim.c
adsptype.c	rt5timer.c
ode0.c	rt_bkgnd.c
ode1.c	rt_log.c
ode2.c	rt_mon.c
ode3.c	rt_trig.c
ode4.c	rtwlog.c
ode5.c	simulink.c
rt5access.c	userfunc.c
rt5event.c	
rt5gett.c	
rt5main.c	
rt5ode.c	

及び *MODEL*_UFUNC.c

７．３．　長時間に渡るDSPの専有

バックグランド処理はアイドルループを回りながら、RTMON からのアクセスに応答する傍ら実行されます。RTMONからのアクセスとは、例えばゲイン値変更とか、変数表示等をさしています。
従って、極端な例では、ユーザ定義のバックグランド処理で無限ループを構成すると、RTMON からのアクセスに応答が出来ず、RTMONからはDSPが止まったか、に見えてしまいます。

無限ループを構成しないまでも､出来るだけ短い時間で､一旦関数から戻る様な記述をしてください。
例として、ユーザ定義関数のバックグランド処理で、ある条件が整うのを待機する様な処理を記述したい場合、以下の様に、条件が整っていなければ一旦関数から戻り、次回呼び出された時再度継続する様な記述としてください。

バックグランド処理記述の例

```
static int  flg=0;
void UserBackground( )
{
        switch( flg ){
             case 0:  if( ! ConditionOk( ) )          /* 条件が整わなければ */
                               return;                /* 一旦戻る */
                        else    flg = 1;              /* 整ったらフラグをセットし */
                        return;                       /* 一旦戻る */
              case 1:   NextProc();                   /* 条件が整った後の処理 */
                        flg = 2;                      /* フラグセットし */
                        return;                       /* 戻る */
              default:  return;                       /* 以後何もしない */
        }
}
```

この例の想定は、関数 ConditionOK( ) は条件が整っているか確認する関数で、条件は長時間の渡り整わない可能性があるものとします。この関数は、条件が整ったら TRUE を、それ以外は FALSE を返すものとします。

７．４．　長時間に渡る割り込み禁止

RTモデルでのブロック線図の計算は､タイマ(又は外部からの)割り込みにて計算を駆動しています。
従って、長時間に渡り割り込みを禁止するとブロック線図の計算が停止してしまいます。
出来れば、レジスターにて停止したい割り込みのみに、割り込みを抑止してください。
どうしても割込みを禁止する必要がある場合は、出来るだけ短時間に抑えてください。禁止して良い時間の目安は、ステップサイズ－実行時間（ブロック線図計算に要する時間）以内です。

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2006.09.19	初版

ＡＤＳＰｘｘ４－ｘ４１Ｄ
ＭＡＴＬＡＢ対応ライブラリ
ユーザー定義関数

中部電機株式会社

〒440-0004　　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006. 9　　第１版発行

ＡＤＳＰシリーズ

リアルタイムモニタ・チュートリアル

ＭＡＴＬＡＢ対応ライブラリ

中部電機株式会社

目　　次

１.概要

本書は、ＭＡＴＬＡＢ、ＳＩＭＵＬＩＮＫ（共に米国 The Math Works Inc 社製品）及びＭＡＴＬＡＢ対応ライブラリ「ＡＤＳＰｘｘ４－ｘ４１」(中部電機(株)製品) を用いて、実時間アプリケーションを作成する具体的手順を、実例を元に説明しています。
本書で説明している各製品のバージョンは以下の通りです。

ＭＡＴＬＡＢ等	7.1 版
ＭＡＴＬＡＢ対応ライブラリ	5.04.01 版
ＤＳＰ用コンパイラ	ＡＤＳＰ６７４用ＣＣＳ３．０版

注）マニュアル記載の MATLAB 図は、Ver7.1 を使用しています。

２.用語・略語の説明

本書では説明の簡略化の為、次の用語（略語）を使用する場合があります。
ディレクトリフォルダと同じ意味

ＤＬＧ	ダイアログボックス
ＭＬ	ＭＡＴＬＡＢ
ＭＬ－ＬＩＢ	ＭＡＴＬＡＢ対応ライブラリ
ＲＴＷ	Ｒｅａｌ-Ｔｉｍｅ Ｗｏｒｋｓｈｏｐ
ＲＴＭＯＮ	リアルタイムモニタ（本製品に含まれる）
ＲＴモデル	リアルタイムモデル
ＳＬ	ＳＩＭＵＬＩＮＫ
ＭＷ	Metrowerks （コンパイラ製造元）
ＴＭＷ	The Math Works Inc. （ＭＡＴＬＡＢ製造元）
ACR0741-00	＠ＣＲＯ７４１

３.モデルの作成

本製品で取り扱える実時間モデルの作成には、２つの方法があります。

３．１．　ユーザプログラミングによるモデル

１つめは、汎用エディターでＣ言語又はアセンブラを用いてプログラミングし、MW社製デバッカを用いてデバッグを行い、実行可能形式にしたものです。本製品では、この形式の実時間モデルについては、ロード・実行はできますが、メモリ内部の値や計算結果のモニタ等を行う事は出来ません。デバッガ等を用いて完全にデバッグを行ったうえ、ロードすればただちに実行できる形式にしてください。

尚、本書ではこの方法については説明を行っていません。

３．２．　ＳＩＭＵＬＩＮＫによるモデル

２つめは、ＳＬにより作成したブロック線図を元にＲＴＷを用いてＣ言語ソースを経て実行形式プログラムを自動生成する方法です。以下本書ではこれをＲＴモデルと呼びます。

例として、＠ＣＲＯ７４１で正弦波を生成し、これをＤＡ変換器に出力すると共に、ＡＤ変換器からデータを取り込むモデルを作ってみます。

３．２．１．　作業用フォルダの作成

ＭＬにはカレントディレクトリという概念があり、作成されたブロック線図や、ＲＴＷで作成するＲＴモデルは全て、このカレントディレクトリに保存されます。この作業用フォルダは、エクスプローラ等のファイル管理ツールで作成します。

３．２．２．　カレントディレクトリの設定

次に、ＭＬを起動します。

ＭＬ起動後、最初に行う作業は、カレントディレクトリを、希望とするフォルダに移動する事です。

ＭＬインストール後、通常に起動すると、カレントディレクトリは C:¥Matlab6p5p1¥Work になっています。MATLAB Command Window に “cd” と入力すると現在のカレントディレクトリが表示されます。

＜例＞

```
》 cd

C:¥Matlab71¥Work
```

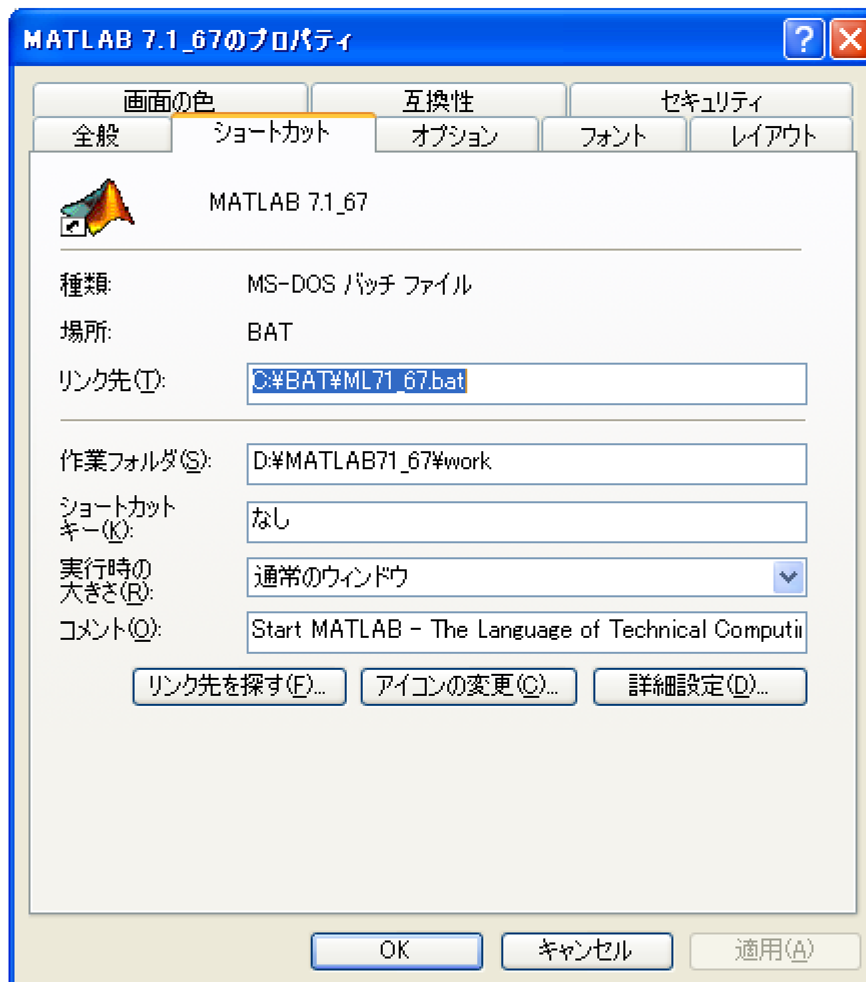
これを例えば、D:¥USER¥SomeOne¥MatlabWork に移動するには、

```
》cd  D:¥USER¥SomeOne¥MatlabWork
```

と入力します。

もし、毎回入力せず、自動的にカレントディレクトリを移動させたいのであれば、例えばデスクトップにショートカットを作成し、このプロパティーのショートカットの作業フォルダ(S)にカレントディレクトリを入力しておきます。

＜ショートカットの設定例＞



ＭＬ起動時のカレントディレクトリを自動的に移動する方法は他にもありますので、どの方法を用いても構いませんが、要点は、保存したブロック線図やデータファイル等を見失わないで済む為、これからＭＬやＳＬで作業をしようとする際には、必ず「カレントディレクトリが何処になっているか？」を確認する事です。

４．モデルの作成

新規モデルの作成は、MATLAB のメニューから、ファイル(F) → 新規作成(N) → モデル (M) を選択します。作成されたブロック線図は内容が空で MATLAB により暫定的な名前（例えば Untitled）が付けられています。名称を決定するには、ブロック線図メニューから、File(F) ～ Save As(A)で名称を付けて（例えば test 等）保存します。拡張子は省略すれば、自動的に .mdl が追加されます。この時、フォルダはＭＬカレントディレクトリになっていますので、変更しないようにします。

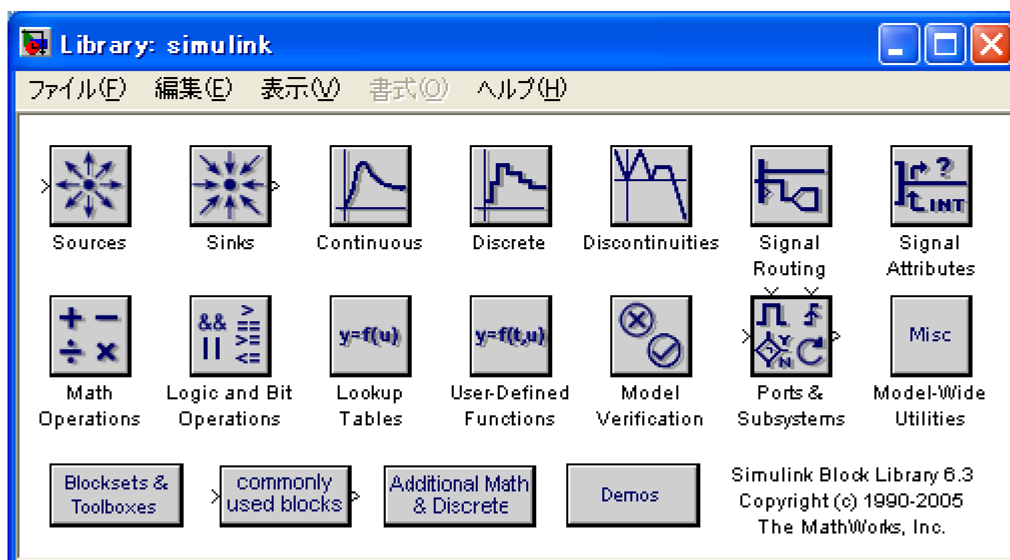
ファイル名称は、ＲＴＷで使用する場合、制約がありますので次の制約を守って命名してください。

数字で始まってはいけません。__ 又は アルファベットで始めます。
（例：_model.mdl、model.mdl＝OK、1model.mdl＝ＮＧ）
かな、漢字は使用してはいけません。
（例：abc_123.mdl＝ＯＫ、ｶﾅ.mdl、漢字.mdl＝ＮＧ）
__以外の記号は使ってはいけません。
（例：abc_123.mdl＝ＯＫ、abc$.mdl、abc%&.mdl＝ＮＧ）
大文字小文字は、同じと見なされます。

こうして保存したブロック線図は、次回ＭＬ起動時に、MATLAB Command Windw にブロック線図の名称を入力する事により、開く事ができます。

続いて作成したブロック線図に適宜ブロックを貼り付け、線で繋いでシステムを構築していきます。ＳＬの標準ブロックは、MATLAB Launch Pad による利用方法と、MATLAB Command window に simulink と入力してSimulink ライブラリブラウザｰを開き､Simulink を選択してマウス右クリックからSimulink ライブラリを開くを選択します。次の要な、ＳＬの標準ブロックライブラリが開きます。

＜ＳＬ標準ブロックライブラリ＞



更に、Sources 等のブロックアイコンをダブルクリックすると個々のブロックを収録したライブラリブロック線図が開きます。

LaunchPad からの利用と、ブロック線図形式の表示からの利用は、本質的には同じです。

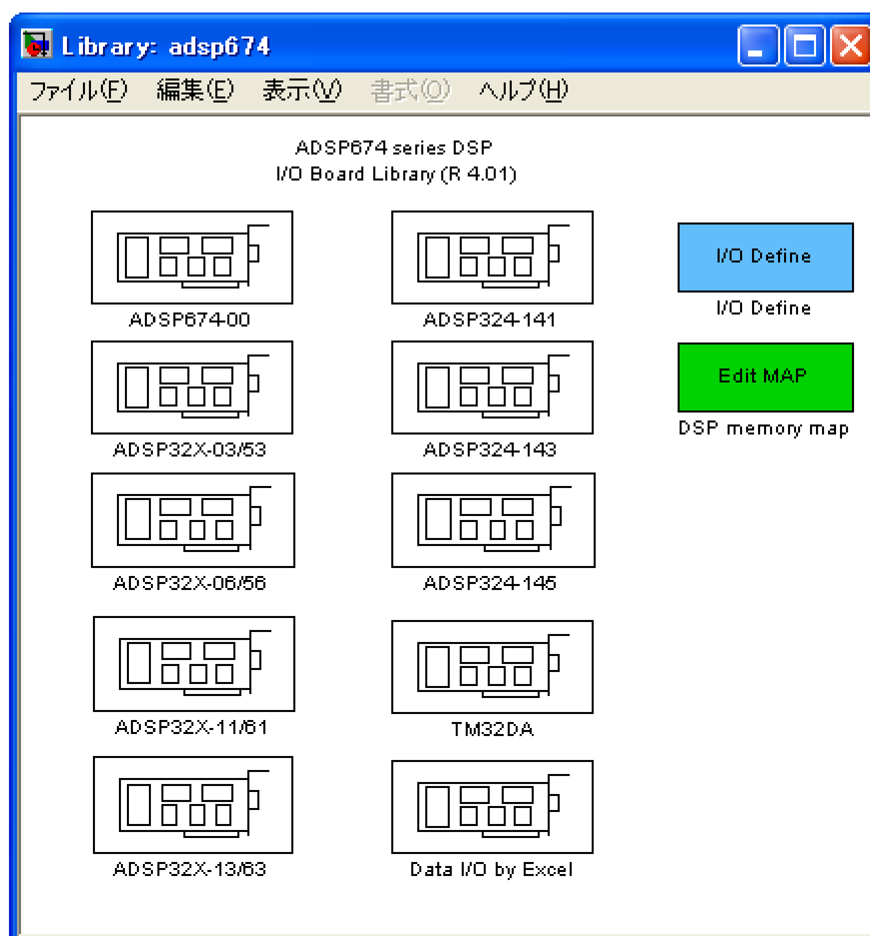
尚、本書の説明では後者の方法でブロック線図を作成する事とします。

次に、ＭＬ－ＬＩＢのライブラリを開く為、MATLAB Command Window に対して、コマンドを入力します。

ADSP674 の場合　　　adsp674
ADSP324 の場合　　　adsp32x

これにより、次の様なライブラリウィンドウが開きます。（下の例は adsp674 の場合です）

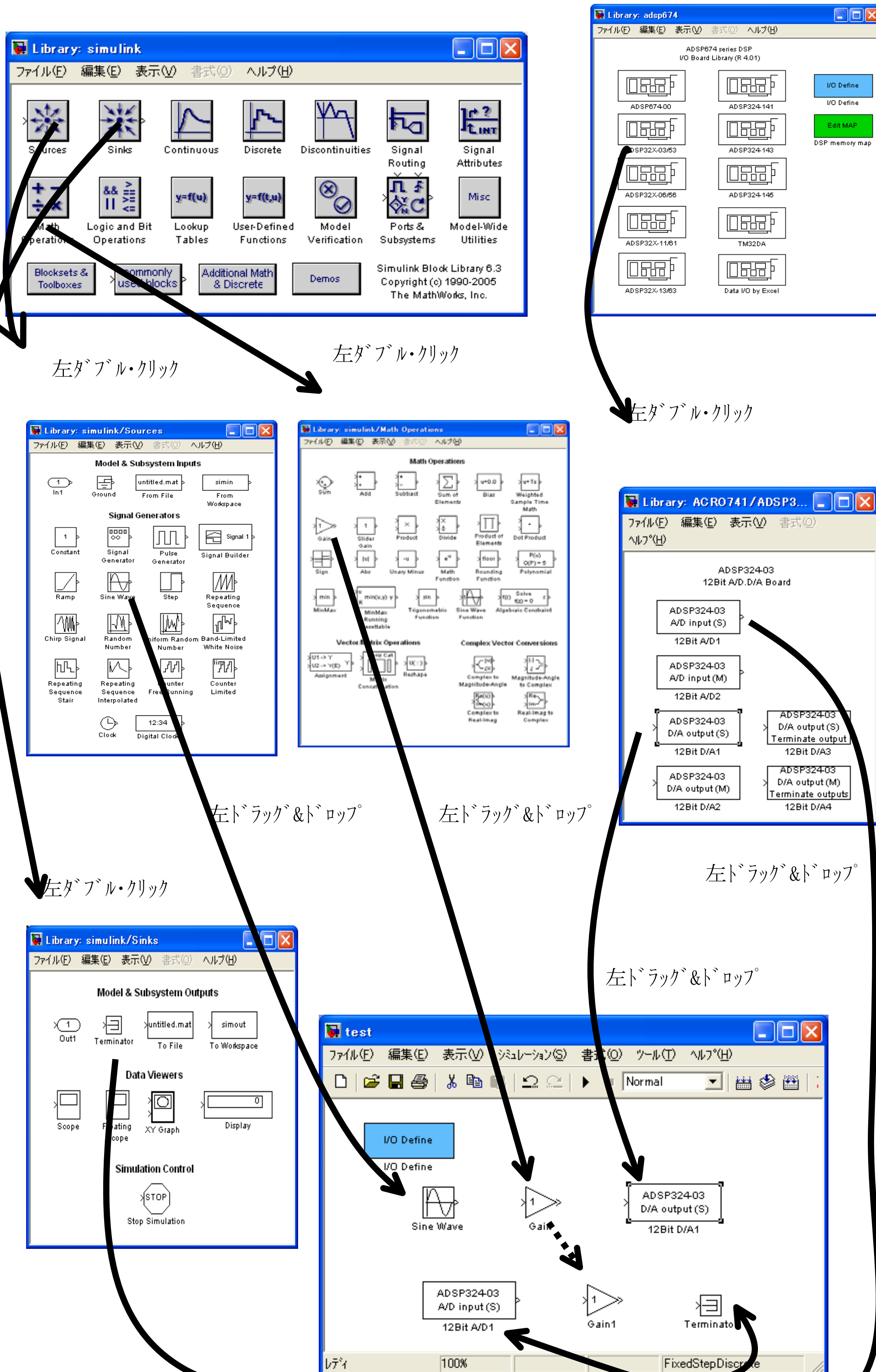
＜ＭＬ－ＬＩＢ ブロックライブラリ＞



開いたライブラリにはライブラリをボード毎に分類して収録したブロックアイコンが有りますので、これをダブルクリックし、開いたブロックライブラリの中から必要なライブラリをドラッグし、ブロック線図にドロップします。

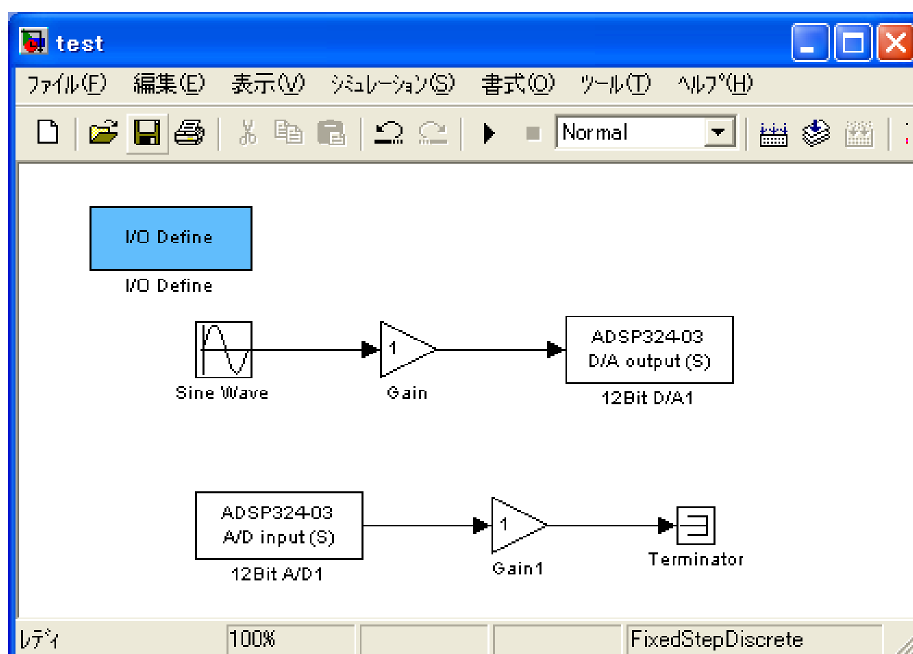
ここでは例として、正弦波をＤＡ変換機に出力すると共に、ＡＤ変換器からデータを入力するブロック線図を作成してみます。

正弦波は、ＳＬブロックライブラリの中の Sources に入っていますので、まず Sources をマウス左ボタンでダブルクリックし、開いたライブラリの中から Sine Wave をマウス左ボタンでドラッグし、ブロック線図にドロップします。以下同様に次ページに示すブロック
をドラッグ＆ドロップしてください。



次に、ブロック線図に貼り付けたブロックの出力から次のブロックへの入力へと、マウス右ボタンでドラッグし結線します。

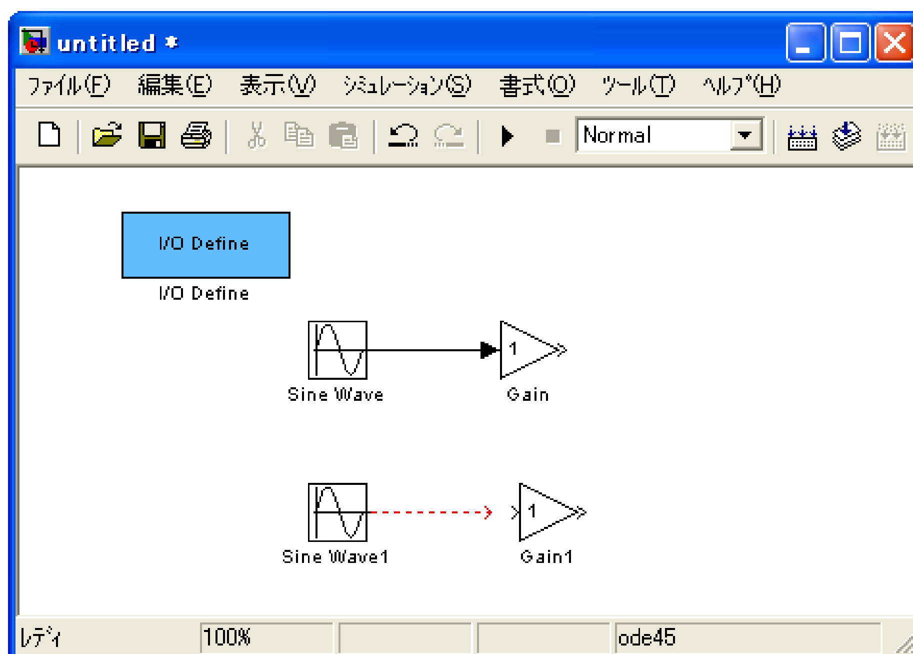
＜結線が終わったブロック線図の例＞



結線が正しく接続出来ている線は ━▶ の形になっています。

次の例は、上側のブロック（Sine Wave と Gain の間）は入力出力共に正しく結線出来ていますが、下側（Sine Wave1 と Gain1 との間）は入力出力共に正しく結線出来ていません。ブロックと線の端が重なっているので、一見つながっているかに見えますが、線の形でつながっていない事が判ります。

＜結線不良（下側）の例＞



この場合は線を一旦削除して引き直すか、又はブロックアイコンをマウス左でドラッグして一旦位置を移動し、正しく結線出来た後元の位置に戻します。

結線が出来たら、パラメータを設定します。ここでは、I/0define ブロックの設定をします。このブロックは実装する、I/0 ボードの枚数を定義します。I/0 の実装がなくても全てゼロ枚で定義が必要です。このモデルでは ADSP324-03 を１枚実装します。

＜I/0define 定義例＞

Define Number of I/O Board

I/Oﾎﾞｰﾄﾞ実装枚数設定

ADSP324-03	1	OK
ADSP324-06	0	ｷｬﾝｾﾙ
ADSP324-11	0	
ADSP324-13	0	標準設定呼出
ADSP324-141	0	標準値設定
ADSP324-143	0	
ADSP324-145	0	

ADSP324-145詳細設定

	標準製品の場合はこちらからお選びください	特注製品の場合はこちらからお選びください
1枚目	----	CNT CPT PIO PLS PWM EMP
2枚目	----	CNT CPT PIO PLS PWM EMP
3枚目	----	CNT CPT PIO PLS PWM EMP
4枚目	----	CNT CPT PIO PLS PWM EMP

正弦波の周波数とＡＤブロックのサンプルタイムを変更します。

Sine Wave ブロックアイコンをダブルクリックすると下の初期設定ダイアログが開きますので、周波数［単位はラジアン／秒］を設定します。例えば１０Ｈｚであれば、pi＊2＊１０を設定し、Close ボタンを押します。

＜Sine Wave 初期設定ＤＬＧの例＞

Block Parameters: Sine Wave

Sine Wave

正弦波の出力。正弦波ﾀｲﾌﾟは計算手法によって決定します。2つのﾀｲﾌﾟで利用されるﾊﾟﾗﾒｰﾀは以下の関係があります。:

周期毎のｻﾝﾌﾟﾙ = 2*pi / (周波数 * ｻﾝﾌﾟﾙ時間)

ｵﾌｾｯﾄｻﾝﾌﾟﾙ数 = 位相 * 周期毎のｻﾝﾌﾟﾙ / (2*pi)

もし長時間の実行で数値的な問題(例えば、絶対時間のｵｰﾊﾞｰﾌﾛｰ)がある場合はｻﾝﾌﾟﾙﾍﾞｰｽの正弦波を利用ください。

ﾊﾟﾗﾒｰﾀ

正弦波ﾀｲﾌﾟ: Time based

大きさ: 1

ﾊﾞｲｱｽ: 0

周波数 (rad/sec): 2*pi*10

位相 (rad): 0

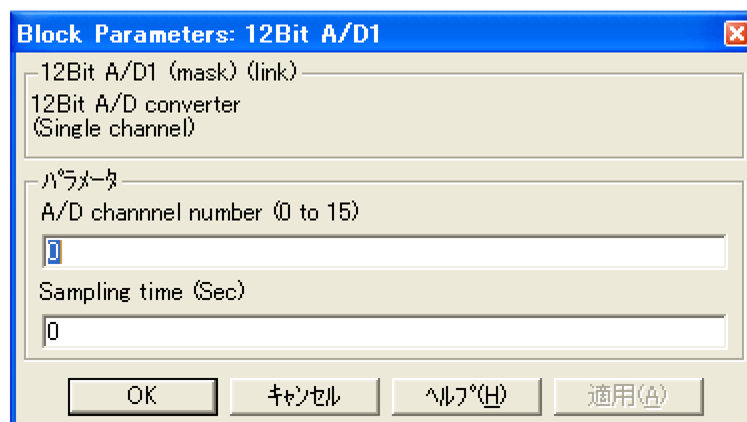
ｻﾝﾌﾟﾙ時間: 0

☑ ﾍﾞｸﾄﾙﾊﾟﾗﾒｰﾀを1-Dとして解釈

OK　ｷｬﾝｾﾙ　ﾍﾙﾌﾟ(H)　適用(A)

ＡＤ変換器の初期設定ダイアログも同様に開き、下記の様にサンプルタイムを０に設定します。サンプルタイム０は連続時間系を意味し、ステップサイズ毎にＡＤからデータを取る事を意味します。

＜ＡＤ変換器の初期設定＞



尚、ＡＤとＤＡはこの例では共にＣＨ０を使います。これ以外のＣＨの場合や、複数のＣＨを使う（複数のブロックを使う）場合は適宜ＣＨ番号の設定が必要です。

設定が出来たら、ブロック線図のメニューから、File 〜 Save でブロック線図を保存します。

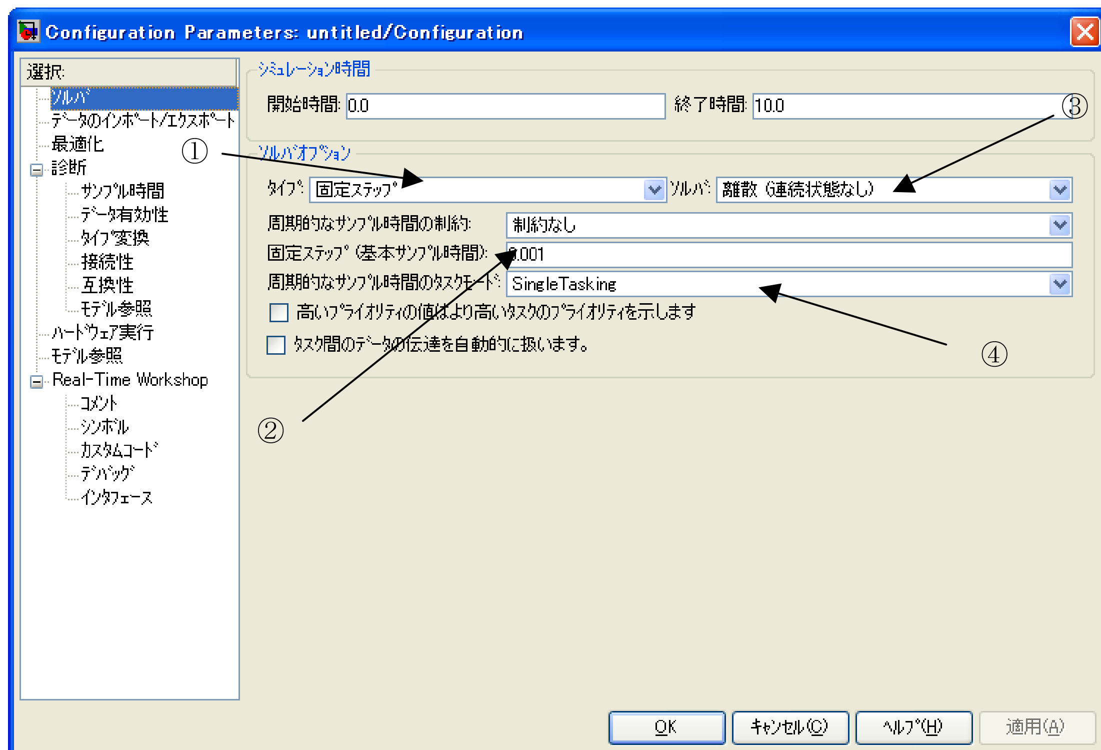
４．１．１．　　モデルの評価

　モデルが希望通り動作するか否かは、ＳＬ上で評価出来る場合と出来ない場合があります。例えば外部機器（制御対象等）と接続しないといけない等、ＳＬだけでは評価出来ないのが一般的ですが、出来れば擬似データ等を用いて予め、ある程度の目処を付けておく事をお勧めします。
　その際に注意が必要なのは、評価の最終段階では、積分アルゴリズムを「固定ステップサイズ」にしても大丈夫か確認する事です。
一般に数値シミュレーションでは、計算精度と計算時間の短縮の為、可変時間刻みのアルゴリズムが良く用いられます。これは、状態変数の導関数を計算し、傾きが急な場合計算精度を上げるため時間刻みを短くし、逆に傾きが緩やかな場合は時間刻みを長くしても計算精度は低下しないので、計算回数を減らして計算に費やす時間を節約する（計算回数を減らす事は同時に誤差の累積を減らす意味をも含む）為、時間刻みを長くします。ＳＬでもこれら、可変時間刻みのアルゴリズムが多数組み込まれています。
しかし、最終目的が「実時間モデルを作る事」の場合、実時間モデルでは可変時間刻みが実現出来ない為、ＳＬで評価の段階で、固定時間刻みで評価をしておく必要があります。固定時間刻みの場合、必然的に時間刻みの幅（ステップサイズ）により計算結果が違って来る場合がありますので、どの程度のステップサイズで計算すれば良いかも確認しておく必要があります。

　固定時間刻みにするには、ブロック線図のメニューから Simulation 〜 Parameters..を選択し次の Simulation parametersＤＬＧを開きます。このＤＬＧの Solver タブを選択し、Solver options グループの Type コンボボックスから Fixed- step を選択します。続いてすぐ下の Fixed step size:に希望するステップサイズを入力します。auto のままではＲＴＷでコード化出来ません。最終的には必ず、ある具体的なステップサイズ値を入力してください。
　又、モデルに連続系の状態変数が無いのに、discrete 以外の積分アルゴリズムを選択した場合は、警告が出ます。コード生成の時も同様に警告されますので、この時点で discrete に変更してください。

次の例は、固定時間刻みで積分アルゴリズム無し（今回使う例 test.mdl では積分アルゴリズムが不要）に設定した例です。

＜シミュレーションパラメータ設定例＞



①ここで固定／可変時間刻みを選択します。Fixed-step が固定時間刻みです。
②ステップサイズを設定します。単位は秒です。具体的な数値を入力してください。
③積分アルゴリズムを選択します。
④シングルタスクを選択します。
まず最初に、①で固定／可変を選択します。これによりダイアログの表示項目が変わりますので、続いて②～④を設定します。

４．１．２．　　コード化

評価が終わったモデルは次の手順でコード化します。
ブロック線図のメニューから、Tools　～ RTW Options.. を選択し、Simulation parametersＤＬＧを開きます。
Solver タブを選択し、Solver optins グループの Type が Fixed-step である事、その右の積分アルゴリズムが希望するアルゴリズムである事、その下の Fixed step size:も同様に希望するステップサイズ値になっている事を確認します。
この値は、ＤＳＰで実行する際の設定とは直接には関係しません（ＲＴＭＯＮではブロック線図の設定とは別に、自由に設定できます）が、ＲＴＭＯＮでＲＴモデルをロードした際にデフォルト設定として呼び出す事ができます。

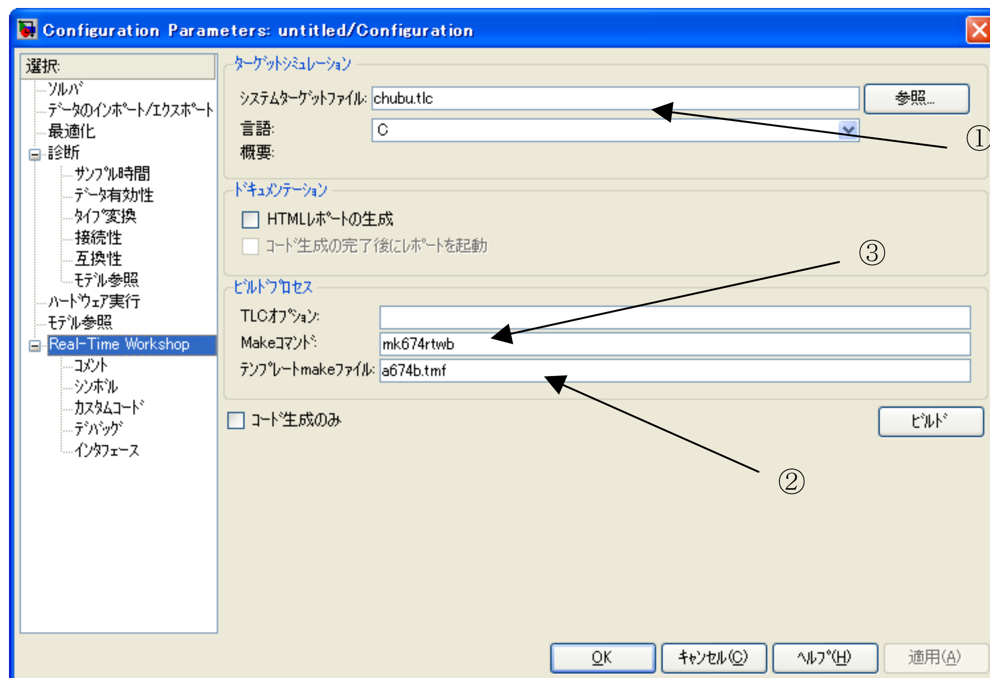
次に RTW タブを選択し、以下を設定してください。
①システムターゲットファイル：chubu.tlc
②テンプレート Make ファイル：

ADSP674 の場合	a674.tmf 又は a674b.tmf
ADSP324 の場合	a32x.tmf

③Make コマンド：

ADSP674 の場合	mk674rtw 又は mk674rtwb
ADSP324 の場合	mk32xrtw

＜ＲＴＷ設定ＤＬＧ設定例＞



この状態で一旦ブロック線図を保存します。そうすれば、万が一ＰＣがハングした場合でも、行った設定はディスクに残っており、次回ＭＬ起動時にブロック線図を開けば設定が復元されます。

注）新規以外で作成されたモデルをコード化する場合、
　　弊社ホームページの「ＭＡＴＬＡＢ対応ライブラリでよくあるＱ＆Ａ」を参照してください。

コード生成の実施は、このＤＬＧのＲＴＷタブを選択し、Build ボタン（図の⑥）を押します。MATLAB command window に以下の表示がなされ、コード生成が完了します。
本製品の版番号や、生成するＲＴモデルの実行形式、及び、初めてコード生成するか２度め以後かにより、表示される行数や内容が異なります。
良否の判断は、全ての行が ### で始まっており（コード生成時のエラーメッセージは###で始まらない)、かつ、最後の 2 行が以下と同様（但し test の部分は個々のブロック線図の名称）であれば、良好と判断してください。

```
### Starting Real-Time Workshop build procedure for model: test
### Generating code into build directory: .¥test_grt_rtw
### Invoking Target Language Compiler on test.rtw
tlc     -r     .¥test_grt_rtw¥test.rtw         'D:¥MATLAB71¥rtw¥c¥adsp32x¥chubu.tlc'
-O.¥test_grt_rtw ・・・

### Loading TMW TLC function libraries


.
Note: Target specific implementation detail has not been specified.  Assuming
the target is the host computer: PCWIN

Bit specification: char=8, short=16, int=32, and long=32

Shift right on a signed integer is implemented as arithmetic shift : true・・・

To override the default specification you must implement chubu_rtw_info_hook.m,
and add it to the MATLAB path.  See the Real-Time Workshop User's Guide or
type 'help example_rtw_info_hook' in MATLAB for more details.


..
### Initial pass through model to cache user defined code
### Caching model source code
.....
### Writing source file test.c
### Writing header file test.h
### Writing header file test_types.h
### Writing header file test_private.h
.
### Writing source file test_data.c
### TLC code generation complete.
### Creating table file test.itb
### Creating linker response file test.lk
### Creating project marker file: rtw_proj.tmw
### Creating test.mk from D:¥MATLAB6p5p1¥rtw¥c¥adsp32x¥a674b.tmf
### Building test: nmake /nologo -f test.mk STEP=0.001 ADD_MDL_NAME_TO_GLOBALS=1
### Compiling test.c
### Compiling test_data.c
### Compiling rt5sim.c
### Compiling rt5main.c
### Compiling rt_log.c
### Compiling rt_trig.c
```

```
### Compiling adsptype.c
### Compiling rt5acces.c
### Compiling rt_mon.c
### Compiling rt_bkgnd.c
### Compiling rt5gett.c
### Compiling rt5timer.c
### Compiling rt5ode.c
### Compiling ode0.c
### Compiling ode1.c
### Compiling ode2.c
### Compiling ode3.c
### Compiling ode4.c
### Compiling ode5.c
### Linking test.elf
### Created
### Successful completion of Real-Time Workshop build procedure for model: test
>>
```

生成されたＲＴモデルの実行ファイルは拡張子.ELF（例：*model*.elf）で、ＭＬのカレントディレクトリのすぐ下にブロック線図の名称に因んだフォルダが自動的に作成され、その中に格納されます。例えばカレントディレクトリがC:¥Matlab71¥Workで、ブロック線図がTestの場合は
C:¥Matlab71¥Work¥test_grt_rtw¥
ＤＳＰ用実行形式ファイル test.out として作成されます。

５．ＲＴＭＯＮの起動

５．１．　ＲＴＭＯＮの起動

　ＲＴＭＯＮは、Ｗｉｎｄｏｗｓのスタートメニューから スタート 〜 すべてのプログラム(P) 〜 MATLAB 対応ライブラリ for ＭＡＴＬＡＢ7．ｘ Ver x.xx.xx 〜 ﾘｱﾙﾀｲﾑﾓﾆﾀｰ(ADSP674 又は ADSP324)を選択して起動します。

６.モデルペアーの作成

６．１．　モデルペアーの概念

これからロードしようとするＲＴモデルがマルチＤＳＰ場合、各ＤＳＰにＲＴモデルを一括してロード出来る様にする為に必要な情報である、何台のＤＳＰを使うか？、どのＲＴモデルを何番のＤＳＰにロードするか？等を定義し、これらの情報をＲＴモデルとは別に一括して保存・管理します。これを「モデルペアー」と呼びます。ＤＳＰを１台しか使わない場合は必要ありません。

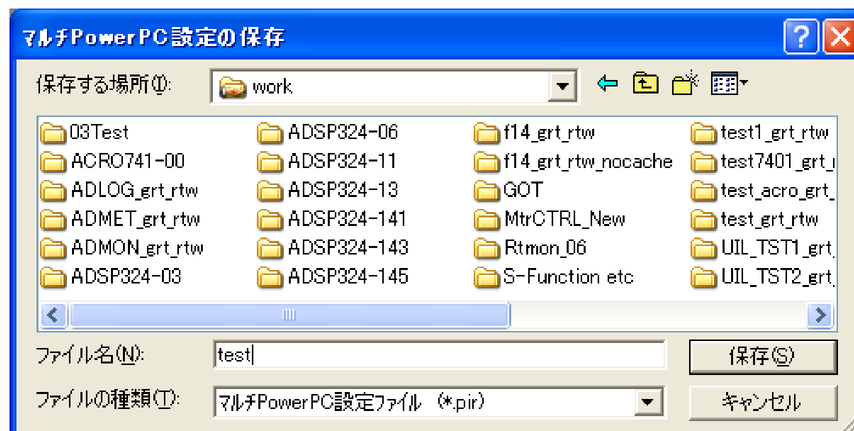
マルチＤＳＰの場合、各ＤＳＰにはそれぞれ１つづつＲＴモデルを割り当てます。ＲＴモデルはブロック線図によるものでも、Ｃ言語やアセンブラで記述したものでも構いません。又、同じＲＴモデルを複数のＤＳＰに割り当てても構いません。

６．２．　モデルペアーの新規作成

ここでは、０、１、２番の計３台のＤＳＰを使い、各ＤＳＰにそれぞれ、test.out、test1.out、test2.elf をロードする場合を想定します。

まず、ＲＴＭＯＮのメニューから、 ファイル(F) 〜 新規ﾏﾙﾁＤＳＰ設定(N)を選択し、マルチＤＳＰ設定の保存ＤＬＧを開きます。既存のファイル名称と重複しないよう気を付けてファイル名称を決定し、保存ボタンを押してください。

＜マルチＤＳＰ設定の保存ＤＬＧ＞



こうして保存されたモデルペアーは拡張子 .PIR ファイルとして保存されます。

ファイル名称を決定しますと、マルチＤＳＰ設定ＤＬＧが開きます。

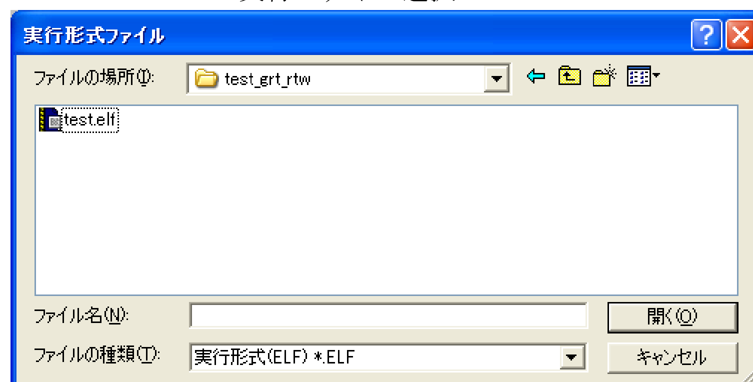
＜マルチＤＳＰ設定ＤＬＧ＞



開いた直後のＤＬＧは全て空になっています。ＤＳＰボード番号の表示が０になっており０番のＤＳＰへロードするモデルが未だ設定されていない事を示しています。

最初に０番ＤＳＰの設定をします。メインモジュールの右の参照ボタン（２個あるうちの上側）を押し、実行ファイル選択ＤＬＧを開きます。このときＤＬＧ左下にある｢相対パス指定｣にチェックを入れておく事により、モデルペアーファイルを基準としたＲＴモデルの相対位置関係を崩さない限り他のディレクトリに移動させてもモデルのロードが可能になります。

＜実行ファイル選択ＤＬＧ＞



ロードしたいＲＴモデル、この場合 test.elf を選択します（上の図）。ＲＴモデルはＭＬのカレントディレクトリに作成されていますので、このＤＬＧでは“ファイルの場所”を使ってフォルダーを間違えない様にしてください。ＭＬとＲＴＭＯＮは直結していませんので、ＭＬでカレントディレクトリをセットしてあっても、その設定はＲＴＭＯＮには直接反映していません。

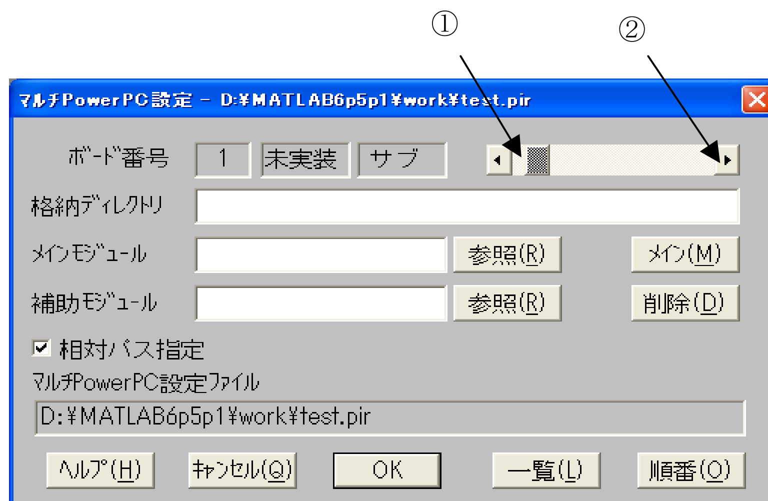
選択したら、開く(0)ボタンを押します。すると、マルチＤＳＰ設定ＤＬＧに今選択したＲＴモデルのディレクトリ名とファイル名の拡張子を除いた部分が表示されます。（下図）

これで０番ＤＳＰへロードするＲＴモデルの設定は完了です。補助モジュールの欄は、通常は空白のままで構いません。

＜マルチＤＳＰ設定ＤＬＧ＞



次に、１番ＤＳＰにロードするＲＴモデルを設定する為、ＤＳＰボード番号を切り替えます。右上のスライダー①をマウス左ボタンで少し右にドラッグするか、又はスライダーの右端にあるボタン②を１回、マウス左ボタンで押します。ＤＬＧの表示は次の様に変わります。



ボード番号の表示が１に変わり、１番ＤＳＰの設定が未だ完了していない事を表しています。先の０番ＤＳＰの場合同様に参照(R)ボタンによりＲＴモデルを選択します。

同様にして０～２番ＤＳＰまで全て設定したら、一覧（L）ボタンを押してＲＴモデルの一覧を表示し、希望通り設定されているか確認してください。

＜モジュール一覧表示ＤＬＧ＞

モジュール一覧

現在設定されているモジュール一覧

OK

番号		モジュール名
0	M	test
1	S	test1
2	S	test2
3	S	-----
4	S	-----
5	S	-----
6	S	-----
7	S	-----
8	S	-----
9	S	-----
10	S	-----
11	S	-----
12	S	-----
13	S	-----
14	S	-----
15	S	-----

もしロードの不要なＤＳＰ番号に誤って設定してしまったら、スライダーでその番号のＤＳＰを選択し、削除ボタンで削除します。

次に、モデルをロードする順番を指定する為、順番（O）ボタンを押します。

＜マルチＤＳＰロード順番ＤＬＧ＞

（未設定）

ﾏﾙﾁPowerPCﾛｰﾄﾞ順番

ロードの順番

先↑ 0 M test ↓後

0 1 2 3 / 4 5 6 7 / 8 9 10 11 / 12 13 14 15 / Clear

ﾍﾙﾌﾟ(H) ｷｬﾝｾﾙ(Q) OK

（設定完了）

ﾏﾙﾁPowerPCﾛｰﾄﾞ順番

ロードの順番

先↑ 0 M test / 1 S test1 / 2 S test2 ↓後

0 1 2 3 / 4 5 6 7 / 8 9 10 11 / 12 13 14 15 / Clear

ﾍﾙﾌﾟ(H) ｷｬﾝｾﾙ(Q) OK

未設定の状態では上図左の様に０番ＤＳＰしか設定されていません。

ＤＳＰ番号で、０→１→２の順にロードしたい場合は、Clear ボタン→０ボタン→１ボタン→２ボタンの順に押します。全てのＲＴモデルがブロック線図による通常のＲＴＭＯＮ用モデルの場合は、若い番号順で良いでしょう。

番号を途中で間違えたら０番から正しい順で再度押して下さい。

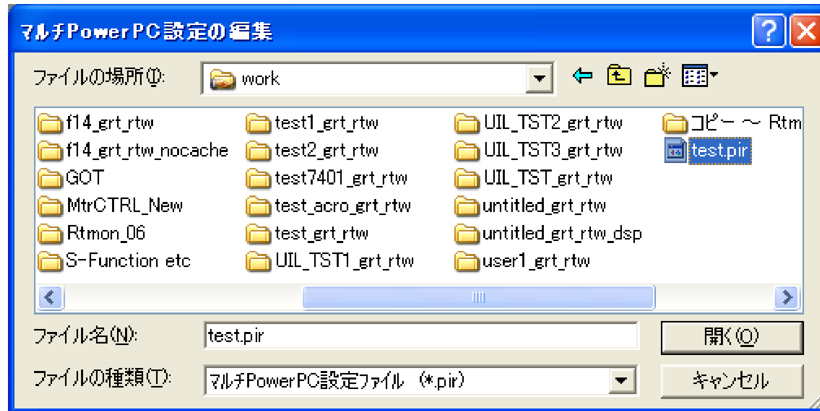
前頁右図、設定完了の様に正しく設定出来たら、マルチＤＳＰロード順番ＤＬＧのＯＫボタンを押します。

６．３．　モデルペアーの編集

既に作成してあるモデルペアーのファイルの内容を変更する場合は、ＲＴＭＯＮのメニューからファイル(F) ～ 既存マルチＤＳＰ編集(E) を選択します。

マルチＤＳＰ設定の編集ＤＬＧが開きますので、ここで編集したいモデルペアのファイルを選択します。

＜マルチＤＳＰ設定の編集ＤＬＧ＞



モデルペアのファイルを選択すると、マルチＤＳＰ設定ＤＬＧが開き、前回保存した際の設定が表示されます。必要に応じて変更を加え、ＯＫボタンで保存してください。

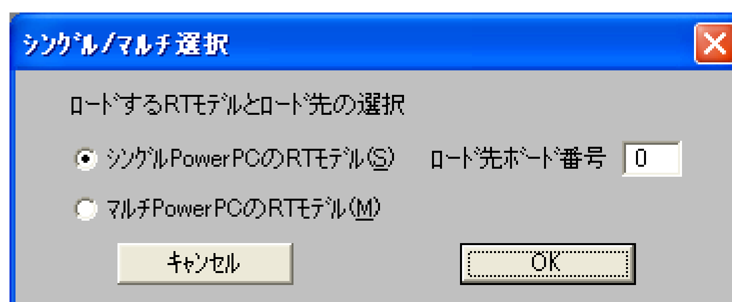
＜マルチＤＳＰ設定ＤＬＧ＞



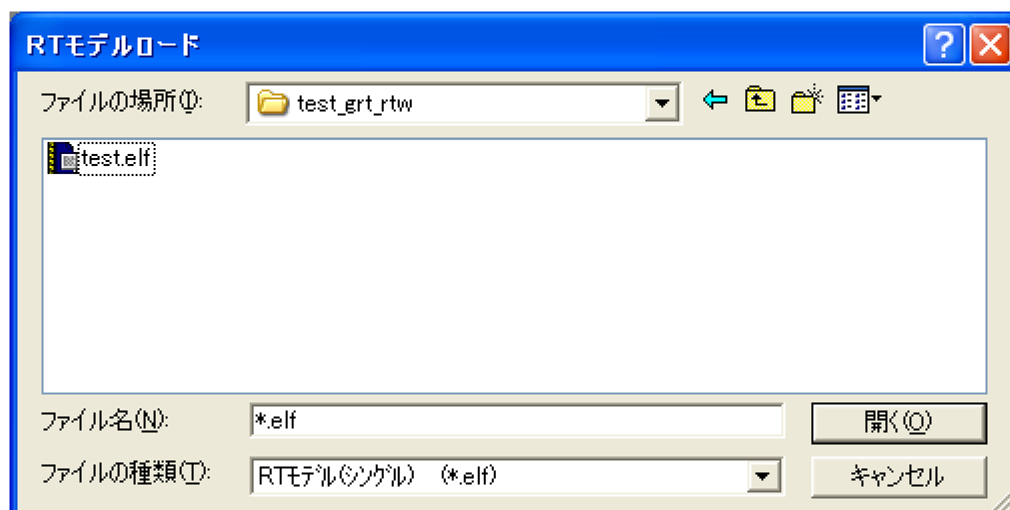
７.モデルのロード

７．１．　シングルＤＳＰモデルのロード

ＲＴモデルのロードはＲＴＭＯＮのメニューから、ﾌｧｲﾙ(F) ～ RT ﾓﾃﾞﾙﾛｰﾄﾞ(O) を選択します。
ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧが開きますので、ｼﾝｸﾞﾙＤＳＰの RT ﾓﾃﾞﾙを選択し、更にロードしたいＤＳＰボード番号を設定(エディットボックスをマウス左ボタンでクリックし､キーボードで DEL キー、BS キー、数字キー尚等を使ってボード番号を直接設定) してください。(下図) 続いてＯＫボタンを押してください。
ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければＯＫボタンだけで結構です。



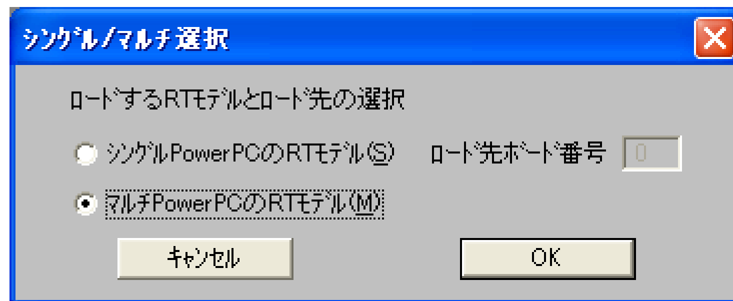
RT ﾓﾃﾞﾙﾛｰﾄﾞＤＬＧが開きますので、ロードしたいＲＴモデルのファイル名を選択し、開くボタンを押してください。



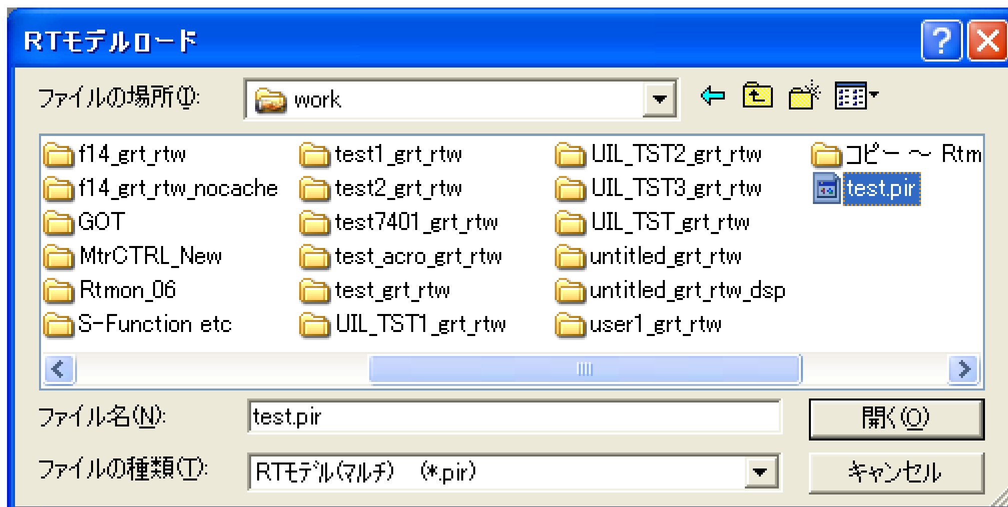
これによりＲＴモデルがロードされ、モデルウィンドウが開きます。

７．２．　　マルチＤＳＰモデルのロード

ＲＴモデルのロードはＲＴＭＯＮのメニューから、ﾌｧｲﾙ(F) ～ RT ﾓﾃﾞﾙﾛｰﾄﾞ(0) を選択します。
ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧが開きますので、ﾏﾙﾁＤＳＰの RT ﾓﾃﾞﾙを選択します。（下図）続いてＯＫボタンを押してください。
ｼﾝｸﾞﾙ／ﾏﾙﾁ選択ＤＬＧはロードの度に開きますが、設定は前回の設定が再現されます。変更が無ければＯＫボタンだけで結構です。



RT ﾓﾃﾞﾙﾛｰﾄﾞＤＬＧが開きますので、ロードしたいモデルペアーのファイル名を選択し、開くボタンを押してください。

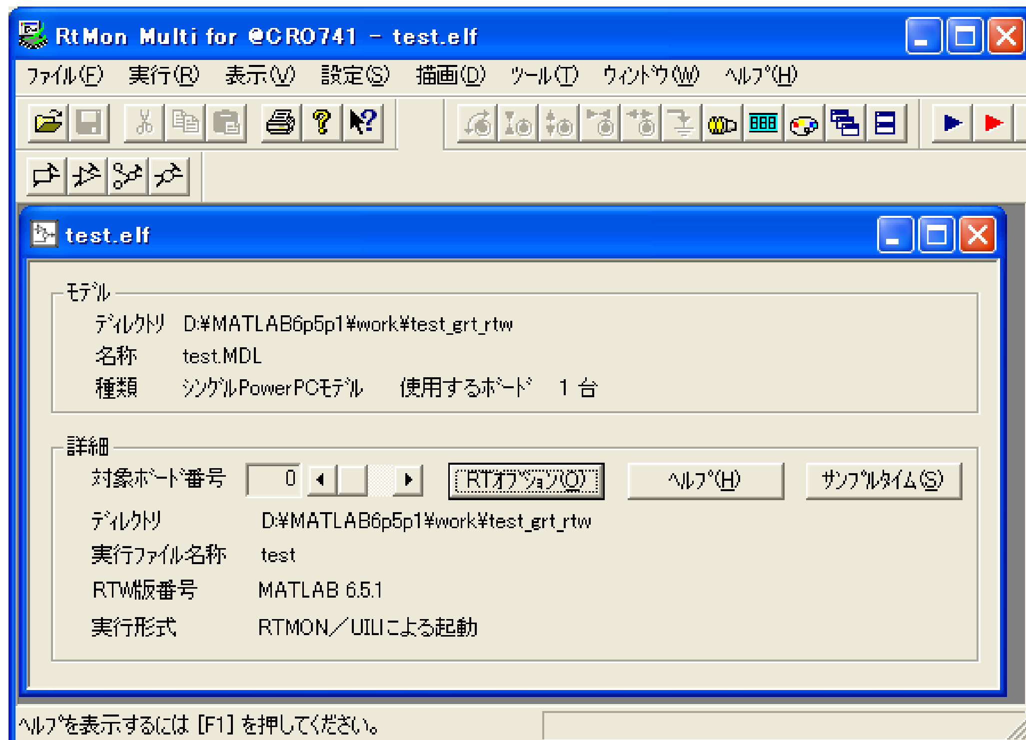


これによりＲＴモデルがロードされ、モデルウィンドウが開きます。

８．実行パラメータ（ＲＴオプション）変更

積分アルゴリズム、ステップサイズ等、ＲＴモデル計算の実行パラメータは、実行に先立ちＲＴＭＯＮから決定します。ＲＴＭＯＮでの設定はブロック線図の設定を反映させる事もできますが、それとは別にＲＴＭＯＮ上で一時的に異なるパラメータで実行する事もできます。それには、元のブロック線図を変更する必要はありません。

変更は、モデルウィンドウ（下図）にある RT ｵﾌﾟｼｮﾝ(0) ボタンでリアルタイムオプションＤＬＧを開きます。



注：この図のアプリケーションの中のチャイルドウィンドウがモデルウィンドウと呼ばれるウィンドウです。この図では、モデルウィンドウを見易くする為、それ以外の表示を隠蔽したうえ、大きさを調整してありますので、通常ＲＴモデルをロードした直後とは表示が異なります。

シングルＤＳＰモデルの場合は使っているボード番号がモデルウィンドウに表示されていますので、そのまま RT ｵﾌﾟｼｮﾝ(0) ボタンを押します。

マルチＤＳＰモデルの場合、設定は個々のＲＴモデル毎にありますので、モデルウィンドウで対象ＤＳＰ番号を選択してうえ、 RT ｵﾌﾟｼｮﾝ(0) ボタンを押します。対象ＤＳＰ番号はその右にあるスライダーで選択します。

１個のＲＴモデルには１個のＲＴオプションしか存在しませんので、シングルＤＳＰモデルとマルチＤＳＰモデルと同じＲＴモデルを使う場合、例えばシングルＤＳＰモデルとしてロードした際の設定を変更すると、マルチＤＳＰモデルの同じＲＴモデルに、その変更が反映されます。

リアルタイムオプション(test200601.MDL)
積分オプション
アルゴリズム None RTW設定(A)
ステップサイズ 0.1 Sec RTW設定(S)
開始時刻 0
終了時刻 10
時間倍率 1
ステップオーバー時の動作
停止
続行
2 回連続で停止
データロギング
Time(T) States(X) Scope
Valiable Output(Y) To WorkSpase
間引き 1 回に１回ﾛｷﾞﾝｸﾞする
ﾊﾞｯﾌｧｻｲｽﾞ 2 K点(×1024)
ﾌｧｲﾙ形式 Binary
ﾌｧｲﾙ名 test200601.MAT 確認
フォルダ変更
OK
ｷｬﾝｾﾙ(Q)
ﾍﾙﾌﾟ(H)
ﾃﾞﾌｫﾙﾄ
登録(S)
呼出(L)
From Excel
ﾌｧｲﾙ名 Test2006b.xls 確認 frmxlsﾌﾞﾛｯｸ使用

リアルタイムオプションの設定画面です。積分オプションを初めモデル実行に際してのオプションを設定します。

アルゴリズムと、ステップサイズは、その右にあるＲＴＷ設定ボタンを押すとＲＴＷでコード生成した時点のブロック線図の設定が呼び出されます。呼び出されるのは、ＲＴＷでコード生成した時点の設定である事に注意してください。ブロック線図の Simulation parameters の設定を変更しても、ＲＴＷでコード生成しないとその変更内容はＲＴモデルには反映されません。

ブロック線図の設定と異なる設定で動作させたい場合、アルゴリズムはコンボボックスのリストの中から使いたいアルゴリズムを選択します。ステップサイズはその時間値［単位：秒］を直接入力します。指数形式（1.5e-003）または、実数形式（0.0015）いずれでも構いません。

開始時刻は、通常は０のままとしてください。

終了時刻は、一定時間ＲＴモデルを動作させたら自動的に停止させたい場合、その時刻値［単位：秒］を、それ以外は充分に大きな値を設定します。（終了時刻に０を設定すると、終了時刻判定は実行されません。）

時間倍率は通常は１としてください。

データロギング条件も設定できます。

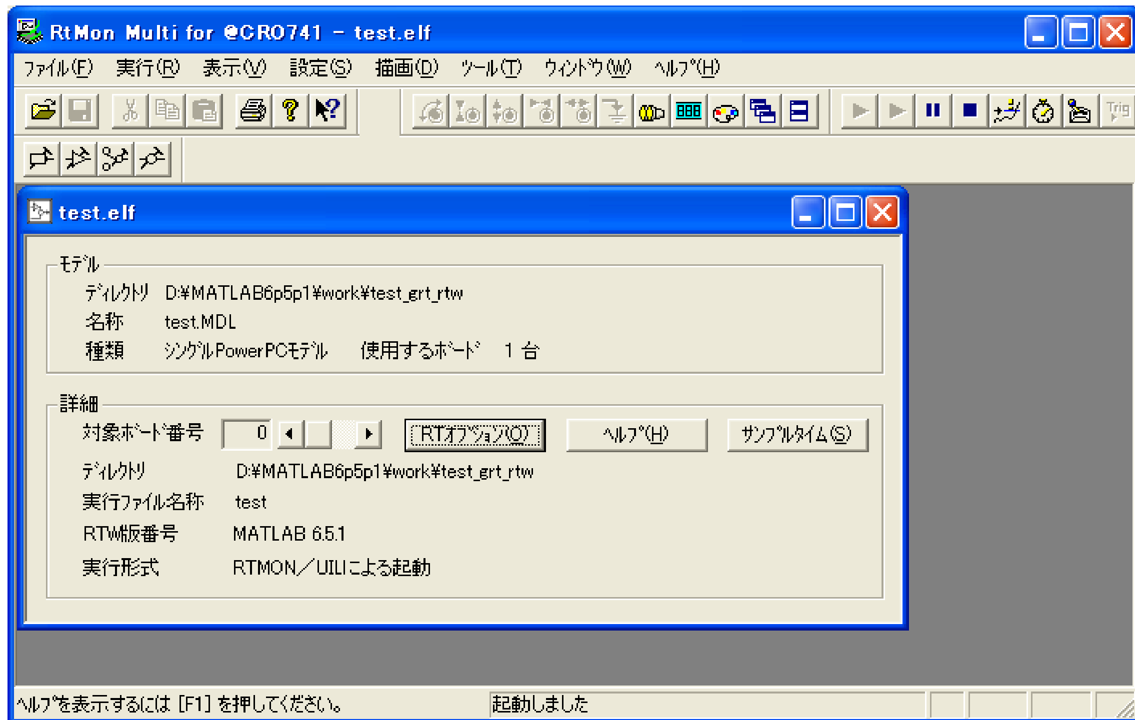
また、モデルにて frmxls ブロックを使用した場合使用する有無と Excel ブックの指定をしてください。ここで指定すると実行前に Excel より@CRO へデータ転送が実行されます。

設定が完了したら、ＯＫボタンを押してください。変更内容はディスクファイルに保存され次回ロード時には同じ設定が自動的に再現されます。尚、ここで設定した値はＤＳＰ上でのＲＴモデルの実行に採用されますが、ブロック線図の Simulation parameters には反映されません。

９．実行開始

実行開始は、メニューから 実行 (R) ～ 実行開始(B) (又はツールバーの実行開始ボタン) で行います。

正常に実行が開始された場合は、ＲＴＭＯＮのステータスバー（場面最下段）に「起動しました」と表示されます。

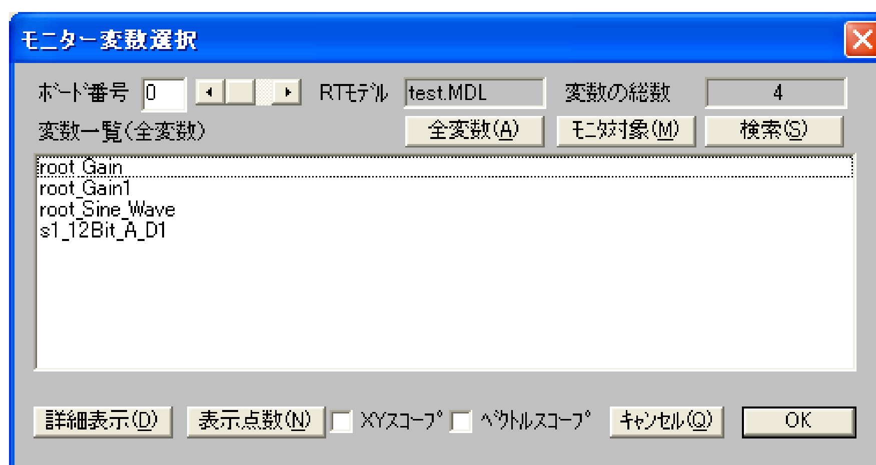


１０．　変数モニター

変数とは、ブロック線図に配置されたブロックから出力された信号を意味します。ブロック線図の表示でいうとブロックとブロックの間の結線に相当します。ＤＳＰのＲＴモデルではＤＳＰ内部のメモリーに割り当てられ、これを表示／ロギングする事ができます。
変数にはそれを出力するブロックの名称が割り当てられます。

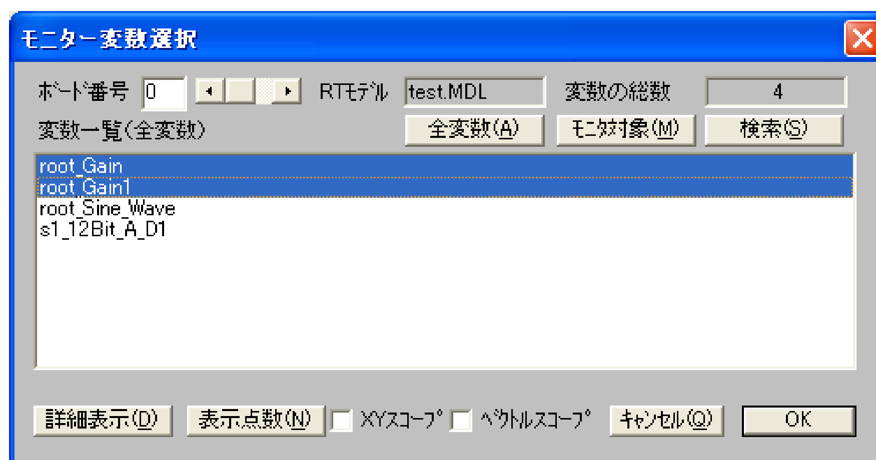
１０．１．　変数の選択

変数モニターはＲＴＭＯＮのメニューから、表示(V) ～ 変数グラフ表示(V) でモニター変数選択ＤＬＧ（下図）を開き、ここでモニターしたい変数を選択します。



モニター変数選択ＤＬＧには、ロードされたＲＴモデルにある参照可能な変数の一覧が表示されます。例えばブロック線図のトップレベルに Gain ブロックが有る場合、この名称を元にした変数名称として root_Gain と表示されます。ブロック線図上の名称と変数名称との関係については、サブシステム・ビューアーのヘルプを参照してください。

ＲＴモデルを作成後、初めてロードした状態では何も選択されない状態となっていますので、このＤＬＧのリストの中から観測したい変数名をマウス左ボタンでクリックし選択状態にします。例えば、トップレベルの Gain1 と Sum の各ブロックの出力を観測したい場合、root_Gain と root_Gain1 を選択します。(下図)



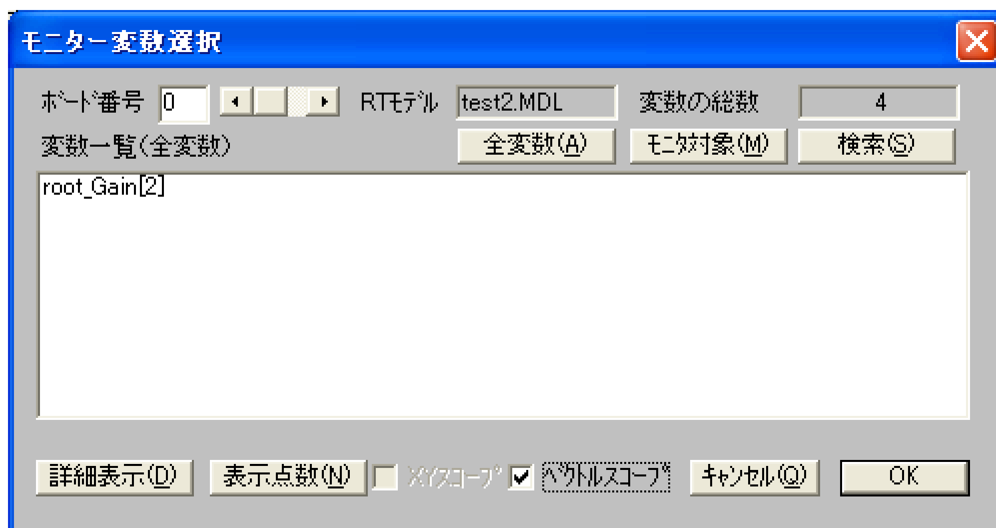
マルチＤＳＰの場合はＤＳＰ番号の右のスライダー又はエディットボックスで目的とするＤＳＰボード番号を選択したうえ、そのボードの変数を選択します。

誤って選択してしまった変数は、再度マウス左ボタンでその変数をクリックすれば非選択状態に戻ります。

選択可能な変数の最大は合計３０個です。

注）観測したい変数名が変数一覧に表示されない場合、弊社ホームページの Q&A「ＭＡＴＬＡＢ対応ライブラリでよくあるＱ＆Ａ」を参照してください。

１０．２．　ベクトル変数の選択



ベクトル情報をもつ変数を観測したい場合は、ベクトルスコープをチェックします。

変数一覧にベクトル情報を持つ変数が表示されるので選択後ＯＫボタンを押してください。

１０．３． 表示点数の設定

選択が完了したら、表示点数ボタンを押し、変数表示点数ＤＬＧを開きます。

変数表示点数

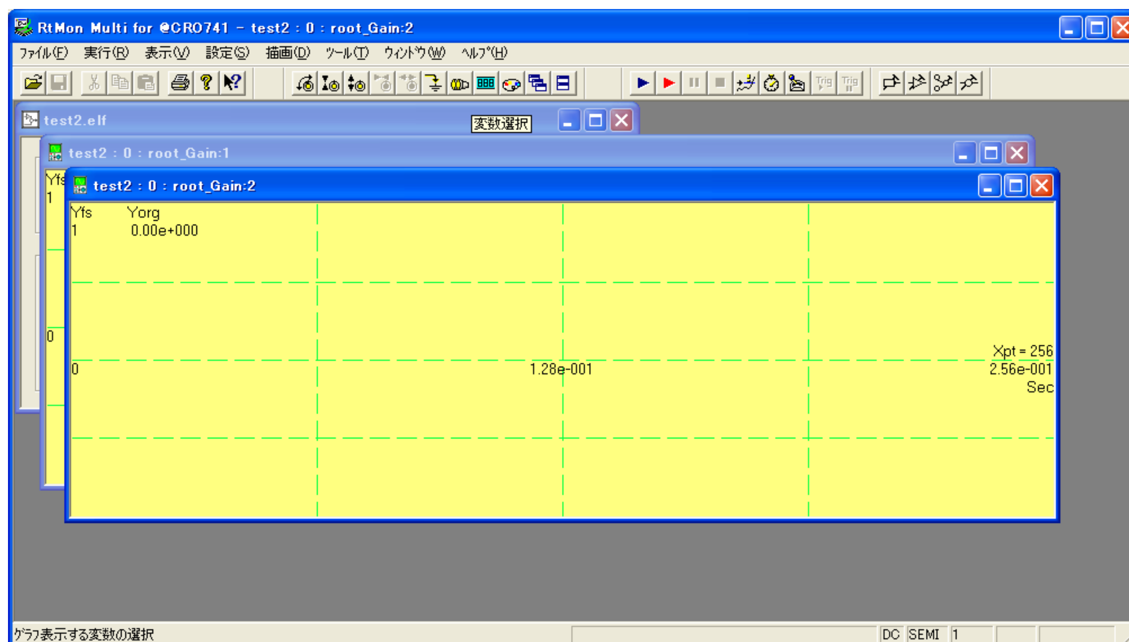
ﾊﾞｯﾌｧのﾃﾞｰﾀ点数	256	点	0.256	秒
表示の全ﾃﾞｰﾀ点数	256	点	0.256	秒
表示の実施点数	256	点	0.256	秒
ステップサイズ			0.001	秒
間引き	1	回に１回ﾛｷﾞﾝｸﾞする		
ﾍﾞｸﾄﾙ用の間引き(時間)	1	回に１回ﾛｷﾞﾝｸﾞする		
ﾍﾞｸﾄﾙ用の間引き(ﾃﾞｰﾀ)	1	点に１回ﾛｷﾞﾝｸﾞする		

ﾍﾙﾌﾟ(H)　ｷｬﾝｾﾙ(Q)　OK

バッファのデータ点数を変更するとその右に表示されている時間が変わります。この時間は、グラフ表示した場合の横軸フルスケール値です。上の例では 0.256 秒となっていますので、グラフ表示の左端から右端までの時間が 0.256 秒となります。これを参照しながらデータ点数を決定します。ただし、間引き設定時は表示時間×間引き数が実際の時間となります。

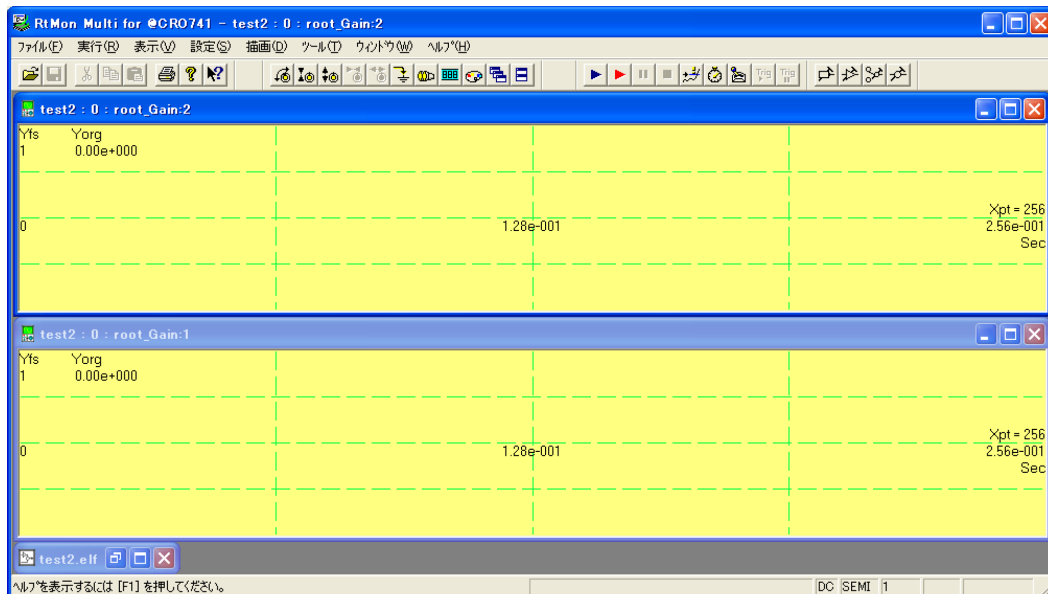
バッファのデータ点数、表示の全データ点数、表示の実施点数は、ここでは共に同じ値に設定します。設定が完了したらＯＫボタンを押してください。

更に、モニター変数選択ＤＬＧのＯＫボタンを押してください。変数選択が完了し、画面には今選択した変数を表示する為のスコープウィンドウが表示されます。



１０．４． スコープの自動配置

開いたばかりのスコープウィンドウは上の図の様に、順次重ねて（カスケード）表示されます。マウスで位置・大きさが適宜変更できますが、綺麗に並べて表示するのが目的なら、メニューから、ｳｨﾝﾄﾞｳ(W) ～ 並べて表示(T) （重ねて表示(C)） 又は、ツールバーの並べて表示 （重ねて表示） ボタンを選択すると、並べて（重ねて）表示するよう自動的に配置されます。また、当面不要な変数はアイコン化したうえで、同操作をすると開いているウィンドウについて再配置されます。下図は“並べて表示”を実施した例です。



１０．５． 変数の数値読み取り

変数をモニターしているスコープウィンドウ内部をマウス左ボタンで左右にドラッグすると、縦方向の線カーソルが表示され、その位置の、画面左端からのデータ点数値と、変数の数値がステータスバーに表示されます。表示位置はステータスバーの右端ですので、ＲＴＭＯＮのメインウィンドウを小さくしている場合は隠れてしまって見えない場合があります。ＲＴＭＯＮを最大化してもう一度試みてください。

変数がまだスコープに表示されていない場合、ステータスバーの数値を表示する場所には（no data） と表示されます。これによりスコープにデータが無いのか、それとも値が大きすぎて画面からはみ出てしまっているのかが判定できます。

１０．６． スコープのレンジ変更

変数の値が大きすぎてスコープ画面からはみ出ている場合､スコープの設定レンジを変更します。

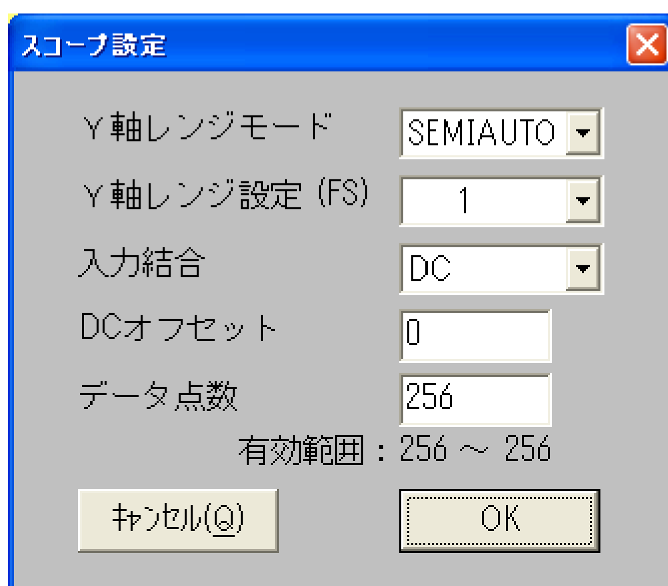
まず、レンジ変更したいスコープウィンドウをマウスでアクティブにします。アクティブか否かは、タイトルバーの色で判断できます。

次に、メニューから、表示(V) ～ 縦軸レンジ拡大(U)（縦軸レンジ縮小(D)）または、ツールバーのＹ軸拡大（Ｙ軸縮小）ボタンを押します。１回押す度に縦軸レンジが約２倍（１／２）に変更されます。縦軸レンジは、１→２→５→１０の様に変更できます。

現在の縦軸のフルスケールは､スコープウィンドウの左端付近にＹｆｓ １ 等として､(及び､ステータスバーにもアクティブなスコープのレンジ値が）表示されています。

１と表示されている場合、縦軸方向の真ん中が０で上端が＋１、下端が－１です。

概ねのレンジ値が予め判っている場合、拡大／縮小ボタンを幾度も押さず、直接レンジ値の設定が可能です。目的とするスコープウィンドウをアクティブにし、メニューから、表示(V) ～ スコープ設定（P）を選択し、スコープ設定ＤＬＧを開きます。



縦軸レンジは“Ｙ軸レンジ設定(FS)”のコンボボックスから選択します。

１０．７． Ｙ軸オフセットを付ける

もし､観測しようとする変数に直流分が乗っている場合、｢拡大しようとすると画面からはみ出してしまう｡かといって､縮小すると波形の細部が見えない｣､といった事態になります｡その場合は、Ｙ軸方向にずらす事ができます。

その方法は３通りあります。

１つ目は、Ｙ軸方向にずらす量（ＤＣオフセット値）を直接設定する事です。設定はＹ軸レンジ設定と同じ、スコープ設定ＤＬＧの“ＤＣオフセット”で行います。例えば、信号の平均値が約０．５の場合、ＤＣオフセットに０．５と入力します。これによりＹ軸方向の真ん中が０．５になります。信号の平均値は、スコープウィンドウをマウス左ボタンでダブルクリックして開いた“スコープ読み取りＤＬＧ”で読み取れます。

２つ目は､マウスを使って視覚的に行う方法です｡まず目的の信号が画面に入るまで縮小します。この時、余り小さくし過ぎず、波形がかろうじて画面からはみださない程度とします。この状態でスコープウィンドウの内部をマウス右ボタンで上下にドラッグします。ドラッグしている間は横方向の線カーソルが表示され、ステータスバーにカーソルの位置の信号値が表示されます。このままドラッグし、波形のＹ軸方向の中心と思しき位置でマウス右ボタンを放します。これにより、ボタンを放した時点のカーソルの位置がＹ軸方向の真ん中に来る様､Ｙ軸方向にグラフが移動されます。これは、ボタンを放した時点のＹ軸値を読み取り、先のＤＣオフセット値に設定したのと同じ意味となります。

３つ目は、自動的に行う方法です。スコープ設定ＤＬＧを開き、入力結合をＡＣに変更します。測定機のオシロスコープのＡＣ／ＤＣ結合のイメージ的には似ていますが、違う点は、実際にハード的にＡＣ結合にするのではなく、データ処理によりＡＣ結合の様な動作をしている点です。この設定の場合スコープは、以下の手順でＤＣオフセットを自動調整します。（１）１回目の表示は現在のＤＣオフセット値でグラフ表示します。（２）表示したデータの平均値を求めます。（３）この値をＤＣオフセット値に設定します。従って２回以上表示を反復する場合に効果が現れ、先回表示したデータの平均値が、画面真ん中に来るよう今回のデータの表示が行われます。平均値が安定している、比較的周期性のある信号の場合に効果が発揮されます。逆に、突発的な現象や、信号変化が緩やかで１周期が横軸の長さより長い場合等は逆効果かも知れません。

１０．８． Ｙ軸オフセットの除去

Ｙ軸方向に移動したグラフを元に戻し、Ｙ軸方向真ん中を数値０にするには、スコープ設定ＤＬＧでＤＣオフセット値を０に設定するか、又はメニューから、表示（Ｖ）～ DC ｵﾌｾｯﾄ除去(Z)を選択するか、又は、ツールバーの表示ｵﾌｾｯﾄ除去ボタンを押します。

１０．９． 変数の削除

不要になった変数モニターを削除するには２通りの方法があります。

１つ目の方法は、不要なスコープウィンドウ右上のＸボタンを押す事です。

もう１つの方法は、モニター変数選択ＤＬＧを開き、不要となった変数をマウスで非選択状態にしたうえ、ＯＫボタンを押す事です。

１０．１０． スコープ状態の保存・選択

現在のスコープ状態が保存できます。通常、初めてモデルロード時は何も表示されません。２回目以降のロード時は前回終了時のスコープ状態がモデル名.scp というファイル名で保存されているのでこのファイルより状態を復帰します。このファイルとは別に現在のスコープ状態を保存・選択できます。

保存･･･　表示→スコープ保存　　ファイル名をつけて保存してください
選択･･･　表示→スコープ選択　　保存したファイル名にて呼び出してください。

１１．　計算所要時間の測定

ＤＳＰ内部でブロック線図の計算にどれだけ時間が消費されているかを知る事ができます。原理的にステップサイズはこの時間より短くする事は出来ませんから、この時間を知る事は、ステップサイズがどこまで短くできるかを知る上で重要な事です。又、時間余裕を知る意味でも重要な事です。

まず、正確な計算所要時間を把握するには、ステップサイズを充分大きくしたうえで、ＲＴモデルを動作させる必要があります。計算所要時間がステップより長いと異常と見なしてＲＴモデルが自動的に停止する仕組みになっている為です。
ＲＴオプション設定のステップサイズを、例えば０．１秒とか１秒等、充分大きい値に設定し、モデルを起動します。

次に、メニューから 表示(V) ～ 計算所要時間表示(N) （又はツールバーの計算所要時間ボタン）を選択し、計算所要時間ＤＬＧを表示します。
尚、ここで計算所要時間が表示できるモデルはＲＴＷで生成したモデルのみです。

計算所要時間

番号	所要時間 [μSec]	ﾋﾟｰｸ	番号	所要時間 [μSec]	ﾋﾟｰｸ
0	2.09	□	8	(未実装)	□
1	(未実装)	□	9	(未実装)	□
2	(未実装)	□	10	(未実装)	□
3	(未実装)	□	11	(未実装)	□
4	(未実装)	□	12	(未実装)	□
5	(未実装)	□	13	(未実装)	□
6	(未実装)	□	14	(未実装)	□
7	(未実装)	□	15	(未実装)	□

OK　ﾍﾙﾌﾟ(H)

この表示の更新の間隔は約１秒ですが、ステップサイズがこれより長いと（例えば１０秒）、正しい所要時間値が表示されるのは、ＲＴモデル起動、又はこのＤＬＧを開いてからステップサイズ（１０秒）経過後となります。

通常、表示値は一定ではありません。ばらつきます。一般のＲＴモデルの場合、ばらつきが数μ秒以内であれば正常と見なして構いません。
しかし、ＲＴモデルがユーザ定義関数等を含んでいて、この関数の実行時間が回を追う毎に変化する場合は、それがこのＤＬＧに反映され、表示値がばらつきます。
また、積分アルゴリズム、データロギングの有無、変数モニタの有無、イネーブルドサブシステムやトリガードサブシステムが有る場合はその実行の有無により、所要時間は大きく変わります。これらの条件を色々変え、最長の値を見極める必要があります。
ＲＴモデルを安全に動作させるには、数値を何回か測定し、その最高値を所要時間と見なし、ステップサイズはこれより長い時間を設定しないといけません。
上記のような所要時間にばらつきがでるモデルについては、“ﾋﾟｰｸ”欄へチェックをして計測してください。実行時からの最長の時間が表示されます。

１２． データロギング

データロギングは、ＤＳＰ内部メモリーにＲＴモデルの動作中のデータを貯え、ＲＴモデルの動作終了後これを取り出しファイル化するメカニズムです。

ロギングデータは、１ステップの計算を行う都度１組みのデータが生成されます。従ってステップサイズＴでＮ秒間動作を継続するとデータはＮ／Ｔ組み生成されます。一方、ＤＳＰに保存できるデータの量にはメモリー容量の都合で上限があります。これを超える事は出来ません。保存しきれないデータは保存領域がいっぱいになった段階で、古い順に捨てられます。

１２．１． ロギング項目

ロギングできる項目は以下の通りです。

○現在時刻値	Time(T)
○状態変数	States(X)
○出力変数	Outputs(Y)
○変数	Variable

現在時刻値は、ブロック線図にClockブロックを付けた場合、このブロックの出力値と同じです。後にＭＬで時刻データが必要になる場合、わざわざブロック線図にClockブロックを取り付けなくても、現在時刻値が保存できます。

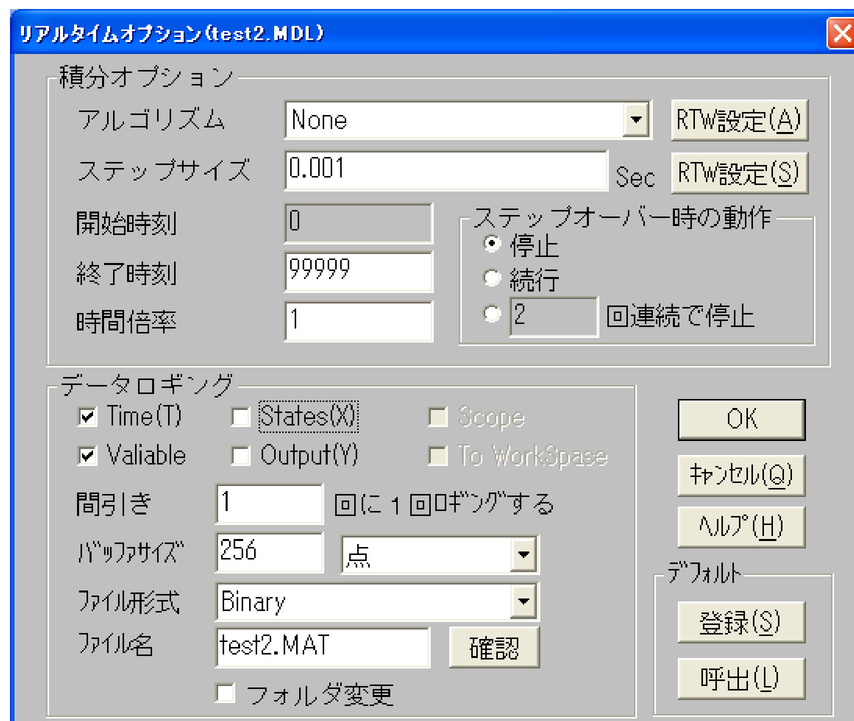
状態変数は、モデルが状態変数を持っている場合、計算ステップ毎の状態変数値が保存できます。但し、ここで保存できるのは、メジャータイムステップ（ステップサイズ間隔）のみです。

出力変数はブロック線図のトップレベルに取り付けた出力ポートに出力されるデータです。これは、ブロック線図から大域変数を経由してユーザ定義関数へ渡されるデータでもあります。

変数は全変数のうち、変数モニターで選択した変数についてロギングされます。

１２．２． ロギング項目選択

上記データロギング項目のどれを取り込むかは、リアルタイムオプションＤＬＧの下半分、データロギンググループのチェックボックスで設定します。



上の例では、現在時刻、変数、が選択されています。

１２．３． データ間引き

長時間に渡りデータを保存したいが、全データを保存するにはメモリー容量が不足する場合、データの間引きを使います。

間引き設定は、１を設定した場合、毎回データが保存されます。２では２回に１回、３では３回に１回のデータが保存され、それ以外のデータは破棄されます。

アンチエリアシング等の処理は行われていません。必要なら、ブロック線図上でフィルターリングを行う必要があります。

１２．４． データ点数

データ点数は、ＤＳＰ内部に確保するバッファの長さと考える事ができます。点数の単位は２種類から選択できます。上の図でバッファサイズの右の、点と表示されているコンボボックスを開きます。選択肢は 点 と K 点(×1024) です。点の場合は、その左に表示されている数値そのものがバッファの点数＝保存できるデータ点数となります。K 点(×1024)の場合は、その左の数字×1024点がバッファの点数となります。例えば、２Ｋ点と 2048 点とは同じ意味を持ちます。

このデータ点数は、ロギング項目毎、変数毎にこの点数分用意されます。

１２．５． ファイル形式・ファイル名

ロギングデータをファイルの保存する場合の形式と名称を設定します。

形式は通常はBinaly としてください。この形式がＭＬのデータファイルと互換があります。ASCIIはＭＬ以外のプログラムとデータ交換する場合用に使用します。ＭＬでは特殊な場合を除きロード出来ません。

ファイル名の拡張子は、Binaly の場合は .MAT 、ASCII の場合は .DAT、CSV の場合は.CSV としてください。

また、“フォルダ変更”にチェック無し時は モデル名_grt_rtw フォルダへ、チェック有り時はモデルのあるフォルダへ保存されます。(MATLAB Version 6.0 以降)

１２．６． マニュアルトリガー

ロギングデータは、モデル実行時に任意の時刻に取得することができます。



ﾄﾘｶﾞｰ ON Trig をｸﾘｯｸするとその時点でデータロギングが停止します。ロギングを再スタートするにはﾄﾘｶﾞｰ有効 Trig をｸﾘｯｸしてください。

１３．　モデルの終了

　ＲＴモデルは次の条件で終了します。
○終了時刻に達した時。
○ＲＴモデルでStop simulationブロックが使われていて、停止条件が成立した時。
○ＲＴＭＯＮで停止操作をした時。
○トリガー機能が有効でかつ自動停止が有効となっており、トリガー条件が成立した時。

ＲＴＭＯＮでは、以下の何れかの操作を行うとＲＴモデルの動作を停止させる事ができます。
１、メニューから、実行(R) 〜 実行終了(E)を選択する。
２、ツールバーの実行終了ボタンを押す。
３、モデルウィンドウのＸボタンを押す。
４、別のＲＴモデルをロードする。
できるだけ１又は２の方法を使用してください。

１４．　ロギングデータの保存

ＲＴモデルでロギングしたデータは、ＲＴモデルの停止後ファイルに保存します。

１４．１．シングルモデルの場合

シングルモデルの場合、上書き保存（予め決めておいた名称で保存）と新規保存（保存の時点で名称を決める）ができます。

１４．１．１．　上書き保存

予め名称を決めておくには、リアルタイムオプションＤＬＧのファイル名エディットボックスで決定します。
上書き保存の実行は、メニューから、ファイル(F) ～ データ上書保存(S)、又は、で保存します。
保存されるデータは、ＲＴモデルが格納されているフォルダーに保存されます。

１４．１．２．　新規保存

上書き保存の実行は、メニューから、ファイル(F) ～ データ新規保存(A)を選択します。



ロギングデータファイル新規保存ＤＬＧが開きますので、ここで既存のファイル名称を確認のうえ、ファイル名(N) に保存したいファイル名称を入力し、保存(S)ボタンを押します。
新規保存ですが、既存ファイル名を指定した場合は警告を表示したうえ、上書き保存されます。

１４．２．マルチモデルの場合

マルチモデルの場合、上書き保存（予め決めておいた名称での保存）しか行えません。
データファイルはＤＳＰ毎に１個ずつ個別に保存されます。これは、複数のＲＴモデルで同じ名称の変数（＝別々のブロック線図に同じ名称のブロック）が存在する可能性がある為、複数のロギングデータを１つにまとめられないことによります。

個々のデータファイルの名称は、リアルタイムオプションＤＬＧで予め設定しておきます。

上書き保存の実行は、メニューから、ファイル(F) ～ データ上書保存(S)、又は、で保存します。
保存されるデータは、ＲＴモデルが格納されているフォルダーに保存されます。

１５．　ロギングデータのＭＬでの利用

　ＲＴＭＯＮで保存したデータをＭＬで利用するには、MATLAB Command Window で load コマンドでロードします。これで初めてＤＳＰで採ったデータがＭＬワークスペースへロードされます。

マルチモデルのロギングデータの場合、各ＤＳＰのデータは個別のファイルになっていますので、load コマンドを反復してロードします。この時注意が必要なのは、各データで同一名称が存在する場合、警告されず、単に後でロードされたデータによって置き換えられる事です。

ロードされたデータの変数名は、以下の通りです。
現在時刻 Ｔ
状態変数 Ｘ
出力変数 Ｙ
変数　変数選択ＤＬＧに表示された名称と同じ。

Ｔはバッファサイズ×１の縦長ベクトルです。
Ｘはバッファサイズ×状態変数の数のマトリクスです。列ベクトルが１つの状態変数の時系列ベクトルです。行方向にどんな状態変数が格納されているかは、現状では簡単に表示する機構が用意されていません。ＲＴＷが出力したソースファイル *model*.h の構造体宣言

```
typedef struct States_tag {
・・・・・・・・
} States ;
```

が参考となります。データはこの構造体のメンバーの順に並んでいます。メンバーが配列の場合はその添え字の０から順に並んでいます。
Ｘのなかの特定の状態変数例えば２行目の状態変数の時系列を取り出し描画するには、MATLAB Copmmand Windowd で

```
X2 = X(:,2); plot(T,X2);
```

と実行します。
Ｙはバッファサイズ×出力点数のマトリクスです。列ベクトルが１つの出力の時系列ベクトルです。Ｘ同様に操作できます。
それ以外のデータも全て列ベクトルです。

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2006.9.14	初版

ＭＡＴＬＡＢ対応ライブラリ
リアルタイムモニタ・チュートリアル

中部電機株式会社

〒440-0004　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006.9　第１版発行

ＡＤＳＰシリーズ

自 動 テ ス ト 説 明 書

ＭＡＴＬＡＢ対応ライブラリ

中部電機株式会社

目　次

１.概要

自動テスト機能は､ＭＡＴＬＡＢ Ｃｏｍｍａｎｄ Ｗｉｎｄｏｗと弊社製リアルタイムモニタ(以下ＲＴＭＯＮ) より起動できる､リアルタイムテストを自動化するソフトウェアです｡実行時に､ステップサイズ／終了時間等の動作条件や、Ｇａｉｎ／Ｃｏｎｓｔａｎｔ／Ｓｗｉｔｃｈ等のパラメータをダイナッミックに変更できます。
また、新たに自動テスト用Ｉ／Ｏブロックとして frmxls / toxls が追加されたことにより、入力データを予め他の計測器等でＥｘｃｅｌファイルとして準備し、実行時にダイナミックに入力できたり、出力データをＥｘｃｅｌファイルとして保存も可能となりました。
このことにより、ＭＡＴＬＡＢスクリプトにて起動コマンドを記述すれば、テスト毎に条件・パラメータを最適化しながら連続テストも可能です。

２.動作環境

1) ＭＡＴＬＡＢ対応ライブラリが動作する環境
2) Microsoft Excel ２０００/ＸＰ/２００３ for Windows が動作する環境

Microsoft は、米国Microsoft Corporation の米国およびその他の国における登録商標です。
ＭＡＴＬＡＢは、米国The Math Works Inc.の登録商標です。

3.実行手順

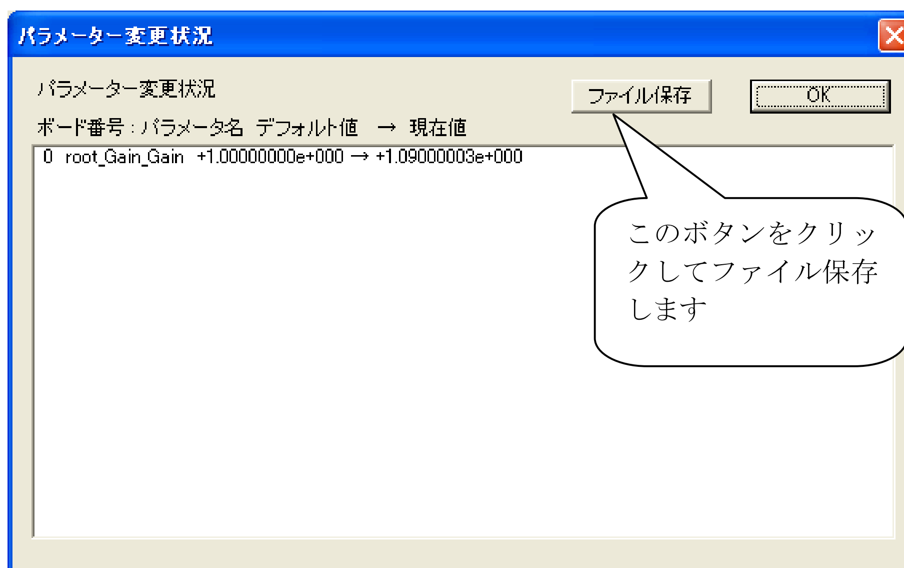
３．１．　モデル作成

ＭＡＴＬＡＢにてモデルを作成し、ビルドします。
自動テスト用Ｉ／Ｏブロックとして、frmxls / toxls ブロックが使用できます。ブロックの使用方法については、“第３章 IO ブロック取扱説明書“を参照してください。

３．２．　パラメータファイル作成

1) ＲＴＭＯＮを起動し、メニュー　→　ＲＴモデルロード　を選択モデルをロードしてください。
2) Ａ）メニュー　→　設定　より設定したい項目（ゲイン・定数・閾値・その他のパラメータ）を選択し値を設定してください。
   Ｂ）弊社製モデルコントロール（別売オプション）の、ボリューム・スライダ・数値入力コントロールからもブロックパラメータの設定が可能です。
3) ③②にてすべて設定が終了したら、　メニュー　→　設定　→　変更状況の表示　を選択し“ファイル保存“にてＴＥＸＴファイルに保存します。

＊その他のパラメータについては、一部実行時に変更できないものがあります。



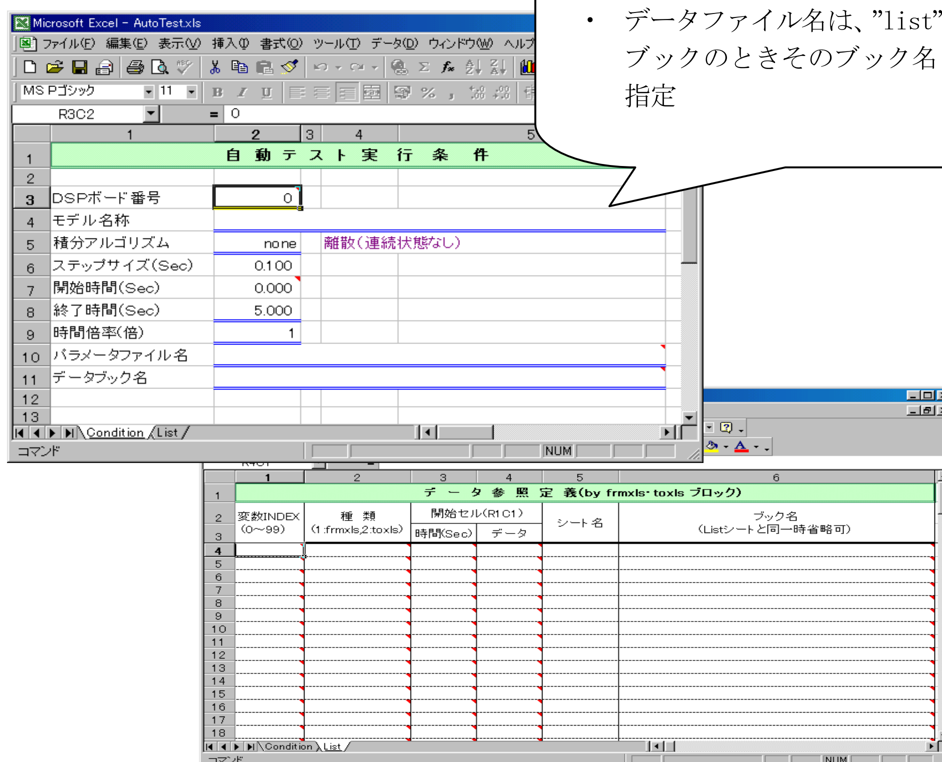
3. 3.　Ｅｘｃｅｌブック作成

下記ディレクトリにテンプレートＥｘｃｅｌＢｏｏｋ“Autotest.xls”が格納されていますので、このブックを利用し動作条件を設定してください。このブックは、“condition”“list”の２種類のシートから構成されています。
また、ブックの作成はＲＴＭＯＮからも可能です。(メニュー→設定→自動テスト用ブック作成)

“condition”…動作条件設定用
“list”　　　…frmxls / toxls ブロックのデータ設定用。frmxls / toxls ブロック未使用時もこのシートは必要です。

<テンプレートＥｘｃｅｌブックディレクトリ>
MATLAB ｲﾝｽﾄｰﾙﾃﾞｨﾚｸﾄﾘ¥rtw¥c¥adsp32x¥rtmon674(rtmonm)

- パラメータファイル名は、2にて作成したファイル名をフルパスで指定。
- データファイル名は、”list”シートが別のブックのときそのブック名をフルパスで指定



４.自動テスト起動

４．１．　MATLAB Command Window 利用

＜コマンド＞

ｓｅｔｏｐｔｎ　…Condition シート：動作条件設定
ｓｅｔｐｒｍ　…Condition シート：パラメータファイル名設定
（ＲＴＭＯＮにて作成したファイルをフルパス指定）
ｓｅｔｄａｔａ　…Condition シート：データブック名設定
(“list”シートのブック名が”Condition”シートのブックと異なるときのみプルパス指定)
ａｕｔｏｔｅｓｔ…自動テスト起動
ｓａｍｐｌｅ　…自動テスト連続起動

上記コマンドを利用して実行します。各コマンドとも終了時には終了メッセージがエラー時には、エラーメッセージとエラーコードが表示されます。エラーの詳細についてはエラーコード表を参照してください。
また、コマンドの引数詳細は、ヘルプを参照してください。

例　help setoptn

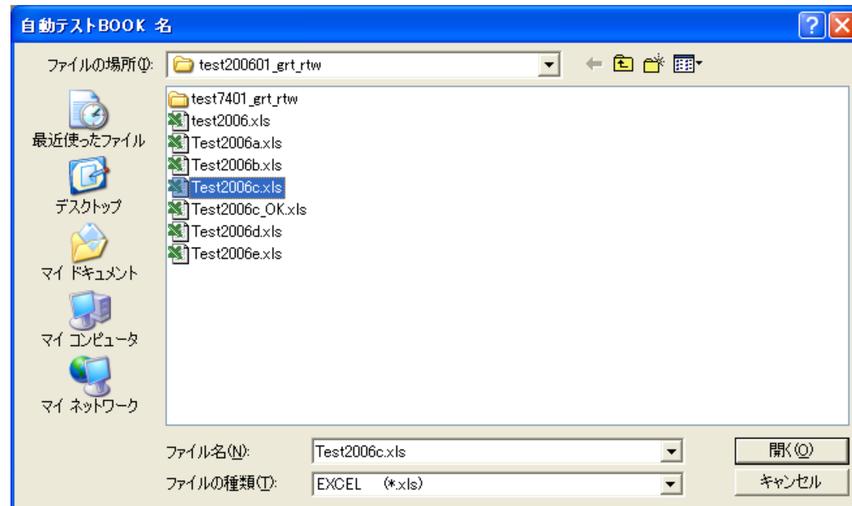
◎コマンド使用例

①>>setoptn(‘autotest.xls’,0.01)　←　step size 0.01Sec 設定
②>>autotest(‘autotest.xls’)　←　自動テスト起動
AutoTest start!<C:¥MATLABR132¥work¥test10_grt_rtw¥autotest.xls> --> Finish!
↑　起動、終了メッセージ表示
④>>setoptn(‘autotest.xls’)　← 現在の設定値表示
⑤>>setprm(‘autotest.xls’, ‘’)　← パラメータファイル名消去
⑥>>sample(‘autotest.xls’,’autotest1.xls’)
↑ 自動テスト連続実行

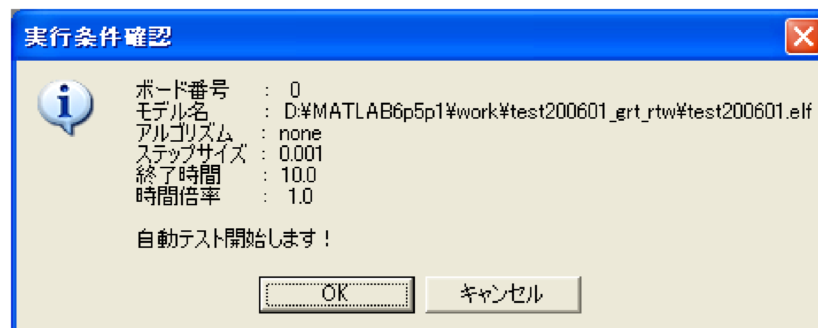
４．２．　RTMON 利用

① ＲＴＭＯＮを起動し、メニュー → 実行 → 自動テスト を選択します。（モデル未ロード時のみ有効）

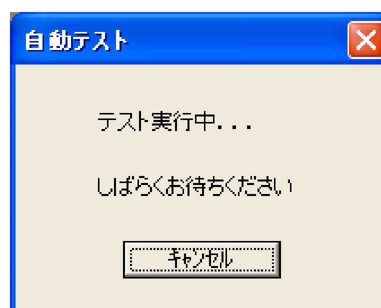
② 下記ダイアログにて計画用Ｅｘｃｅｌ Ｂｏｏｋを選択します。



③ 「開く」を選択すると実行条件確認メッセージが表示されます。内容を確認し応答しください。「ＯＫ」を選択するとテストが開始されます。



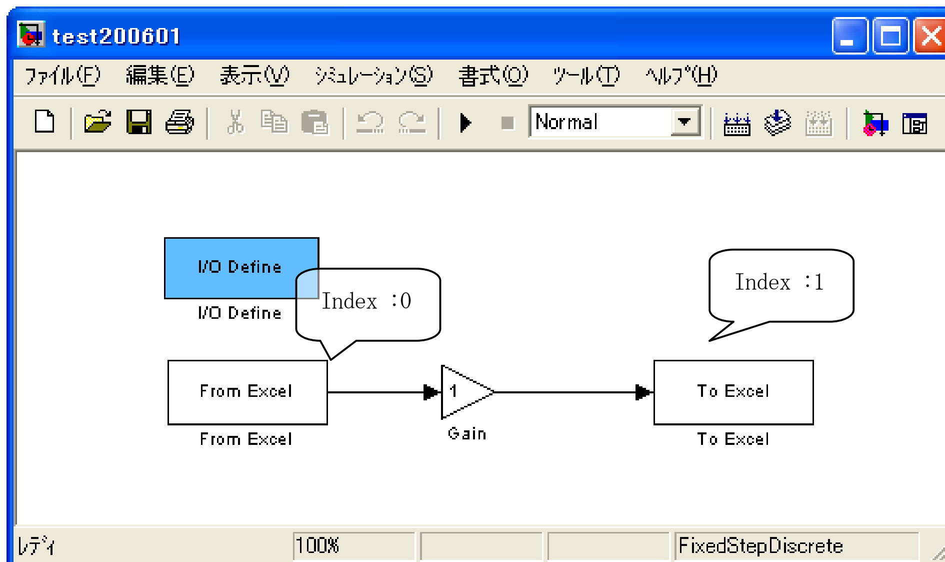
④ 実行中は、実行中ダイアログが表示され、終了するとダイアログが消えます。
実行中に関係するＥｘｃｅｌ Ｂｏｏｋを操作（開く・編集・消去など）するとエラーとなります。



⑤ テスト実行中にエラーが発生するとダイアログメッセージが表示されます。詳細については、５．５．エラーコード表を参照してください。

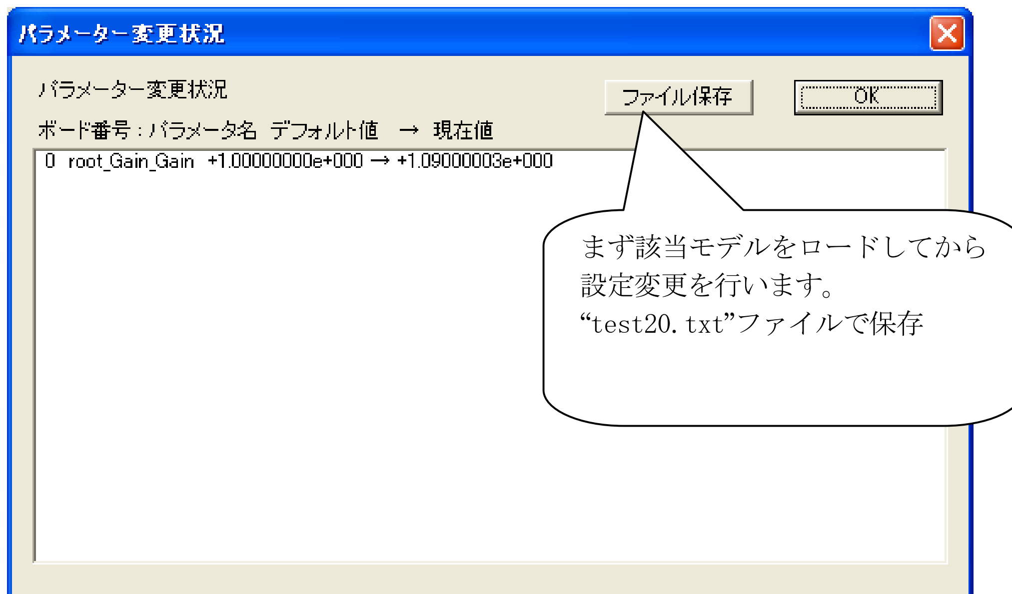
５.実行サンプル

５．１．　モデル作成　→　モデルビルド

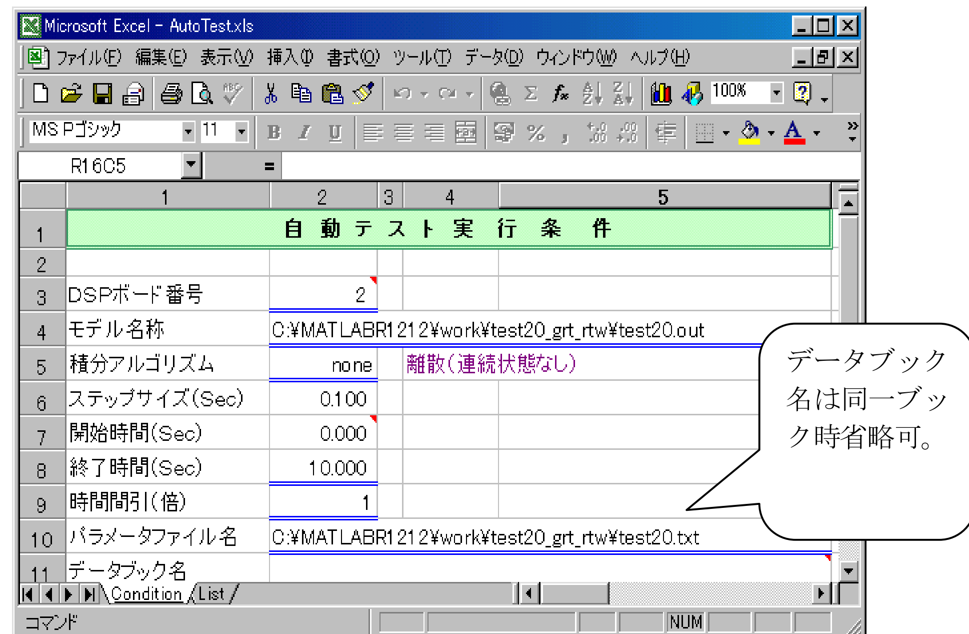


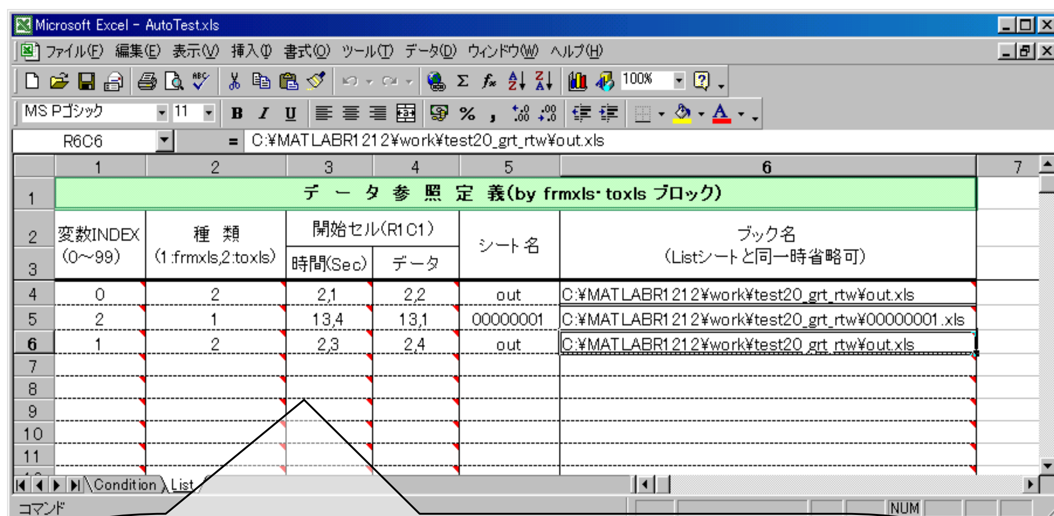
５．２．　パラメータファイル作成　→　ＲＴＭＯＮにて作成

＜３．２．パラメータファイル作成参照＞



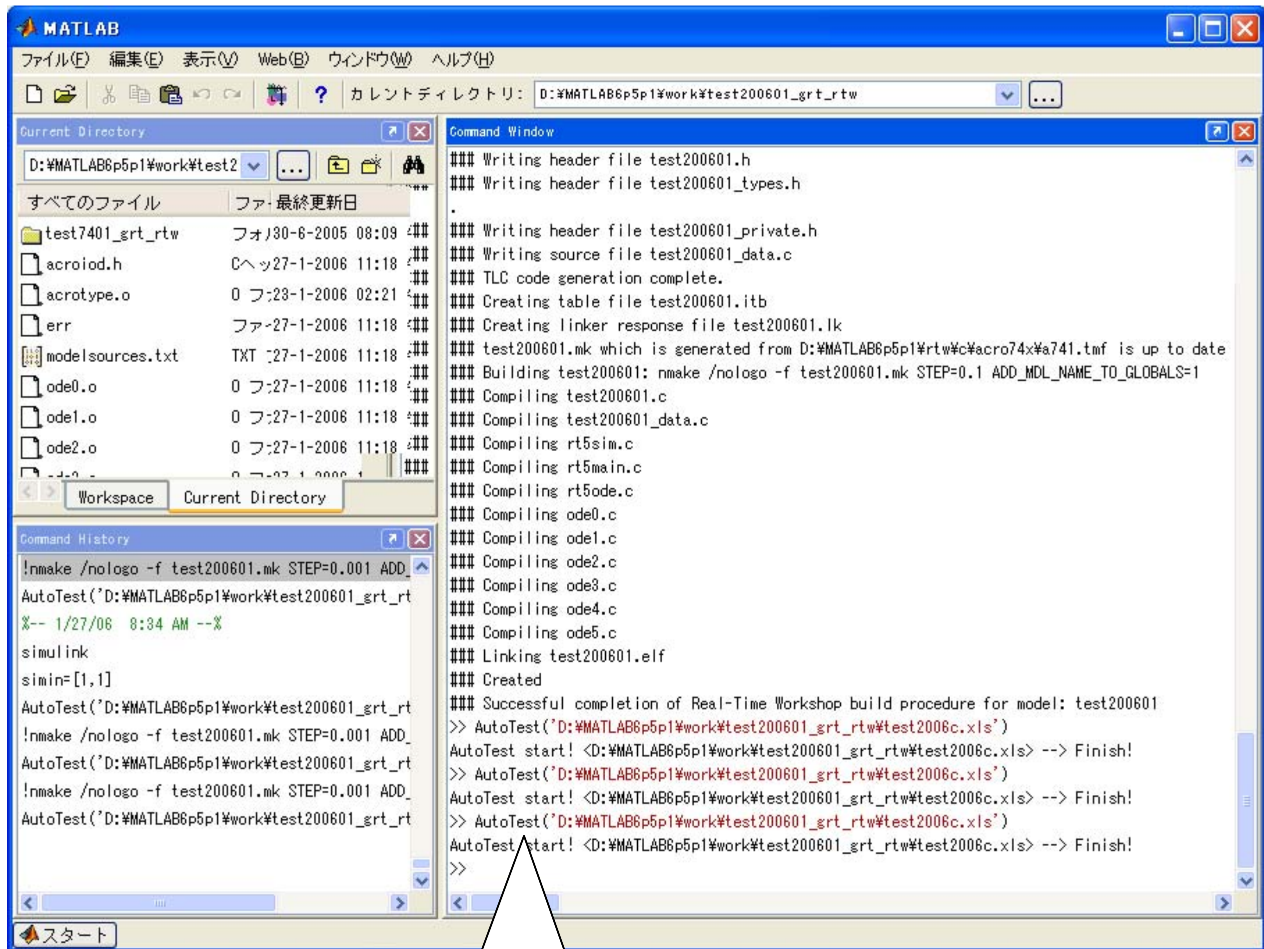
5．3．　ＥｘｃｅｌＢｏｏｋ作成　→　テンプレートブックをコピーして作成





モデルの Variable Index に対応するように情報を設定します。
セル番地は、R1C1 形式にてコンマで区切り入力。
シート・ブックは、toxls ブロックに限り存在しなければ新規で作成されます。
ブック名は、フルパス名で設定します。但し"list"シートと同一のブックの場合は省略可。

５.４.　起動　→　ＭＡＴＬＡＢ　Ｃｏｍｍａｎｄ　Ｗｉｎｄｏｗから起動



>>autotest(‘autotest.xls’)
と入力し起動。
カレントフォルダにＥｘｃｅｌＢｏｏｋが存在する場合、フォルダ名は省略可。

＊Ｅｘｃｅｌブックの名称は任意に変更して使用してください。

５．５．　　エラーコード表

エラー番号	内　　容
１０	AutoTest への引数エラー
２０	Excel アプリケーション起動エラー
３０	IndicationBook Open エラー
３１	IndicationBook ボード番号エラー
３２	IndicationBook モデルエラー
３３	IndicationBook アルゴリズムエラー
３４	IndicationBook ステップサイズエラー
３５	IndicationBook 終了時間エラー
３６	IndicationBook 時間倍率エラー
３７	IndicationBook データ Book エラー
３８	ＤＳＰ初期化エラー
３９	データ Book Open エラー
４０	データ Book Block なし
４１	パラメータファイルエラー
４２	データ転送エラー(frmxls ブロック)
４３	ＤＳＰエラー
４４	ＤＳＰタイムエラー
４５	ＤＳＰその他エラー

【 改 訂 履 歴 】

改訂番号	改訂日付	改 訂 内 容
初版	2006.9.16	初版

MATLAB対応ﾗｲﾌﾞﾗﾘ
自動テスト

中部電機株式会社

〒440-0004　　愛知県豊橋市忠興３丁目２－８
TEL <0532>61-9566　　FAX <0532>63-1081
URL : http://www.chubu-el.co.jp
E-mail : csg@chubu-el.co.jp

2006.9　　第１版発行